JRC SCIENTIFIC AND POLICY REPORTS

Pan-European Risk Management in SDIs

ORCHESTRA Project Report

Nicole Ostlaender, Roberto Lucchi, Michael Lutz,
Anders Friis-Christensen, Ioannis Kannellopoulos,
Angelo Quaglia

Report EUR 24001 - EN

European Commission
Joint Research Centre
Institute for Environment and Sustainability

Contact information
Nicole Ostländer, Michael Lutz & Ioannis Kannellopoulos
Address: Joint Research Centre, Via Enrico Fermi 2749, TP 262, 21027 Ispra (VA), Italy
E-mail: {nicole.ostlaender | ioannis.kannellopoulos}@jrc.ec.europa.eu
Tel.: +39 0332 78 9271 | 5115
Fax: +39 0332 78 6325

http://ies.jrc.ec.europa.eu/
http://www.jrc.ec.europa.eu/

This publication is a Reference Report by the Joint Research Centre of the European Commission.

JRC54181

EUR 24001 EN

ISBN 978-92-79-19091-9 (pdf)

ISSN 1831-9424 (online)

doi:10.2788/37971

Luxembourg: Publications Office of the European Union, 2013

Printed in Italy

Table of Contents

Preface

ORCHESTRA stands for Open Architecture and Spatial Data Infrastructure for Risk Management. As an Integrated Project, ORCHESTRA was partly funded by the European Commission's 6th framework program within the action IST-2002-2.3.2.9 Improving Risk management [IST-2002-51167]. The goal of ORCHESTRA was to design and implement the specifications for a service oriented spatial data infrastructure to improve interoperability between risk management authorities in Europe, and to improve the handling of disaster risk reduction strategies and emergency management operations.

This report describes the outcome of the ORCHESTRA Work Package (WP) 4.2. The following institutions collaborated in the WP 4.2 and thus directly or indirectly contributed to this report. The name in brackets is the name used to refer to an institution in the text:

- Atos Origin (ATOS)
- Joint Research Centre of the European Commission (JRC)
  - Spatial Data Infrastructures Unit (SDIU)
  - Land Management Unit (LMU)
- Fraunhofer IITB (IITB)
- Austrian Research Centers GmbH-ARC, smart systems Division (ARCS)
- Eidgenoessische Technische Hochschule Zuerich (ETHZ)
- Intecs S.p.A., Informatica e tecnologia del software (Intecs)

The project in its entirety is described in the ORCHESTRA Book [1]. Further information and links to further documentation and related publications can be found at the ORCHESTRA homepage[1].

[1] http://www.eu-orchestra.org

1. Introduction and context

Natural disasters are one of the major challenges that trigger cross-border and pan European cooperation in order to protect the environment and the citizens. Due to organisational and technological barriers, actors involved in hazard and risk management often cannot cooperate efficiently. In order to address this shortcoming, the European Commission has made *Improving risk management* one of its strategic objectives in the Information Society Technology (IST) research programme for the 6th framework programme. The ORCHESTRA project [IST-2002-51167], as an *Integrated Project*, stands for *Open Architecture and Spatial Data Infrastructure for Risk Management*. The goal of ORCHESTRA is to design and implement specifications for a service oriented spatial data infrastructure to improve interoperability between risk management authorities in Europe, and to improve the handling of disaster risk reduction strategies and emergency management operations.

The architectural model that underlies ORCHESTRA is a Service Oriented Architecture (SOA). SOA describes an information technology (IT) architectural style, where the logic of business process automation is decomposed into smaller, possibly distributed, units of logic (cf. [2]). The ORCHESTRA Architecture is open, publicly available and free of charge. The architecture's principles are laid out in the Reference Model–ORCHESTRA Architecture (RM-OA) [3]. The RM-OA contains those aspects of the architecture that are independent of the risk management domain. It provides a specification framework for system architects, information modellers and system developers. The framework is a platform-neutral (abstract) specification of the informational and functional aspects of service networks taking into account and evolving out of architectural standards and service specifications, namely of the International Organization for Standardization (ISO), the Open Geospatial Consortium (OGC), the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS). The structure of the RM-OA follows the viewpoints of the ISO/IEC 10746-1 Reference Model for Open Distributed Processing (RM-ODP) (cf. [3]).

The RM-OA shall act as a framework for designing and implementing risk-specific applications. The main objective of four *ORCHESTRA pilots* was to build such risk- and platform-specific applications based on the RM-OAs risk-neutral and abstract specifications and thus test, and, when required, improve the ORCHESTRA architecture and framework. These pilots were based on four real-world application scenarios:

- The first pilot addressed risk and damage assessment of forest fires and flooding. Its main objective was to test the ORCHESTRA architecture within the setting of pan-European risk and damage assessment.
- The second pilot addressed the challenge of planning and management of multi-risk scenarios. It focused on supporting the interaction among different types of end-users, such as modellers, protection bodies, policy makers and administrative managers, for the elaboration of common, collaborative risk maps for several types of natural hazards.
- The third pilot addressed damage assessment in a cross-border setting. It focussed on the road network in the French-Italian border region between Nice and Genoa and on damage caused by natural or man-made disasters.
- The fourth pilot addresses marine environmental risks, focussing on the German Bight. The risk values are calculated by the simulation component that accesses observation data and parameters from several data sources.

This report describes the first pilot on pan-European hazard-assessment in more detail, by describing the application and related use cases, the platform-specific service specifications and application schemas, and the prototypic implementations. For further details on the other thee pilots, such as involved project partners, results and related publications, please visit the ORCHESTRA website[2] or consult the ORCHESTRA book [1].

[2] http://www.eu-orchestra.org/index.shtml

2. Concept of the ORCHESTRA pilot for Pan-European Hazard Assessment

This report describes the ORCHESTRA pilot for pan-European hazard-assessment (PEUNHA). The focus of the pilot lies on the development of applications supporting hazard, damage and risk assessment for forest fires and floods. The pilot was carried out under the lead of the Action Inspire in the Spatial Data Infrastructure Unit (SDIU) of the JRC Institute for Environment and Sustainability (JRC IES). The main stakeholders in the pilot were the following two JRC IES Actions in the Land Management Unit (LMU).

- JRC IES Action INFOREST: The European Commission DG Joint Research Centre set up since 1999 a research group to work specifically on the development and implementation of advanced methods for the evaluation of forest fire risk and mapping of burnt areas at the European scale. The use cases for the forest fire application are based on information provided by INFOREST.
- JRC IES Action WDNH: The WDNH Action, short for 'Floods and other weather-driven natural hazards – Prediction and Mitigation' is developing harmonised EU-wide methodologies and information systems for the prevention and prediction of weather-driven natural hazards to complement national initiatives. Among other tasks, the action is evaluating flood defence and mitigation plans in major European trans-national drainage basins through scenario modelling of the effects of engineering measures, land-use change including regional development (e.g. urban expansion) and climate change effects on flood risk. The use cases for the flood application are based on information provided by WDNH.

Three main user groups are addressed for the activities in the PEUNHA pilot:

- Application developers shall be supported in the creation of SOA-based applications for the assessment of forest fire risk and hazard or damage caused by flooding.
- Domain forest fire experts in the JRC Action INFOREST (who conduct policy support towards various EC DGs) shall be supported in assessing forest fire hazard and risk.
- Interested European citizens shall be supported in assessing the damage caused by (hypothetical) flood events.

All user groups and objectives are described in more detail in use cases in Section 2.2, using use case diagrams in the Unified Modeling Language (UML) (cf. e.g. [4-6]). If ever applied beyond the presented prototypic case, the applications' user groups could possibly be extended as follows: they could cover decision makers within the supported European Commission DGs (DG Environment and DG REGIO) and national decision makers and stakeholders within the national land management authorities, and member states (MS) to create risk maps on flooding or other hazards.

The following main technological challenges were addressed throughout the pilot development:

- Integrating heterogeneous data sets and application schemas;
- Performing distributed geo-processing for hazard and risk assessment though a flexible recombination of services into new risk-specific applications;
- Designing and implementing risk-specific services by means of opaque service chaining;

Each of these challenges addresses one or more aspects of the RM-OA, from testing the suitability and completeness of RM-OA service specifications to the proposed solutions for application development. The challenges were addressed in three different applications that (cf. section 2.1).

2.1. The thematic scope of the three PEUNHA applications

The PEUNHA client is divided into three different pilot applications: 1) thematic application development, 2) forest application and 3) flood application.Together the applications form the PEUNHA pilot and address the user groups and the technological challenges described in the previous section. In the following sections the scope of each of these applications is described.

2.1.1. Thematic application development

The pilot on thematic application development investigated how application developers can benefit from the ORCHESTRA architecture when creating new service-based applications for hazard and risk assessment.

The ORCHESTRA architecture is a Service Oriented Architecture where a set of core functionality is offered through the network. This core functionality is the basis for implementing increasingly specialized services that become part of specific applications for hazard and risk assessment. The core idea is to create new functionalities by instantiating new services that are, where possible, using and combining functionalities offered by the existing services that are part of an ORCHESTRA service network. This is achieved by 1) using discovery services to find suitable existing services and 2) by using workflow languages to chain these services into a sequence of actions that represents the new functionality or application.

Under these premises the ORCHESTRA platform-based applications were developed by implementing a client responsible for the interaction with the users and for triggering the execution of the necessary workflow of service invocations. These workflows are designed as new services, usually called service chain instances, which rely on workflow engines responsible for the entire control and data flow expressed in the workflow description. In this way the execution of the entire workflow is controlled by the client via the invocation of a single service (i.e. the newly developed service chain instance). This type of chaining modality is described as opaque chaining modality in [7].

2.1.2. Forest fire application

Forest fires are particularly affecting the southern European. Each year, the fires cause damage to the environment, infrastructure, economic sectors and private property. [8] provides an overview of the forest fires in Europe for 2005.

Forest fire information at the European level has been collected by the member states since the nineties. This includes information on a number of parameters that describe fire outbreaks: the ignition point, the cause of the fire, the time it was detected and extinguished, the burnt area and many more (see Figure 1 for an example representation of the records as fire statistics per administrative units (NUTS 3 and NUTS 5)).

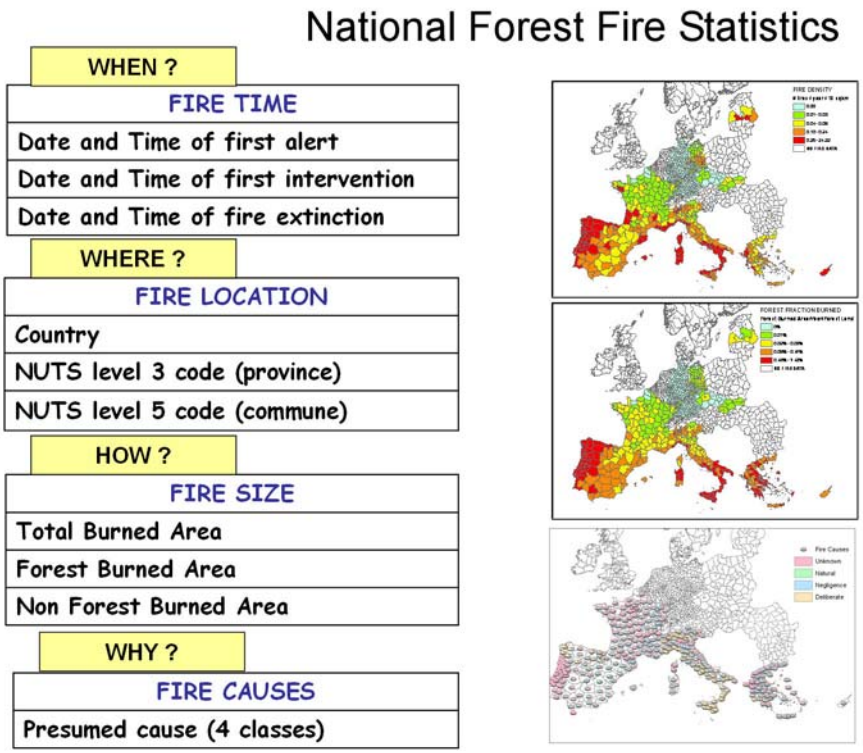


Figure 1: Schematic view on deriving Pan-European Forest Fire Statistics, source: European Forest Fire Information System EFFIS [9].

Analysing this data allows for the creation of forest fire hazard maps (i.e. the likelihood of forest fires), forest fire damage maps, and through their combination, forest fires risk maps. These support decisions on measures for risk prevention on a European scale.

To support this, the pilot provided an application based on fire records for:

- Hazard mapping through forest fire frequency based on the number of fires per administrative units like communes (NUTS 5) regions (NUTS 3) or countries (NUTS 0).
- Forest fire density maps per administrative units describing the normalised fire frequency as fire frequency per km^2.
- Risk mapping per administrative units through the combination of forest fire density (hazard) and burnt area (damage) into risk classes.

The pilot's area of interest for this application covered France, Italy, Spain and Portugal. The time of interest for forest fire recording was 01/01/2003 – 31/12/2003.

2.1.3. Schema mapping

Forest fire information in the different countries is collected in various local schemas. In order to perform a froest fire risk analysis on a European scale, the information on forest fires needs to be harmonised into a common schema, as proposed in the implementation rules to the Forest Focus Regulation (EC No 2152/2003) (for more information on the legal background see http://effis.jrc.ec.europa.eu/about/legal-background). To support this, the pilot provides an application for the registration of mappings between local application schemas to a common, harmonised application schema.

2.1.4. Flood application

Large-scale flood events, like the one experienced throughout Central Europe in August 2002, impact a large number of people and sectors across many national boundaries. Informing the public and raising awareness for damages that might occur are important tasks for decision makers in the field of risk assessment and risk reduction.

To support this, the pilot provides an application for flood simulation and damage assessment. The application allows citizens to create flooding scenarios based on an experimental grid that describes for each grid cell the relative position to the closest river. Based on user input on the flood height, the hypothetical flood extension is simulated. The flood extent is used to identify values like population and land use, which are exposed to the flood. Simple rules are used to derive damage from this exposure, like number of people living in the flooded area. The information is aggregated by administrative units.

The pilot's area of interest for this application covers a subset of the catchment of the rivers Rhine, Moselle, Sarre and Neckar. As the flood is a simulation, time of interest is not explicitly defined, but depends on the acquisition time of the flood exposures data sources.

2.2. Use case descriptions

To describe the three PEUNHA applications and user groups in more detail, UML use cases have been created. A Use Case is a collection of possible scenarios (i.e. sequences of interactions) between a system under discussion (i.e. the application) and its external actors. Use cases offer a systematic and intuitive means of capturing functional requirements by addressing the question, what the system to be developed can do for each of its users. Thus, the driving force for the development process is not a functional specification in the sense of a list of functionalities that system developers consider useful, but a set of requirements that describe the value of the system for its users, from the user's perspective.

Each use case has a central system under consideration, one primary actor and zero or more secondary actors. As a general rule, a system under consideration always interacts directly with all its actors. Therefore, a system might be the 'systems under consideration' in one use case and the 'primary actor' in another, to not violate this rule.

The use cases for the PEUNHA pilot are divided into two groups according to the main actors: use cases for risk and damage assessment by analysts and use cases for application development by software developers:

- **Application development**: The use cases for application development are created for application developers. The developer defines workflows of service operation invocations that are necessary for the risk analysis to be performed as part of a new risk assessment application.
- **Risk and damage assessment**: The use cases for risk and damage assessment are created for analysts, such as domain experts and citizens. They are further subdivided into the two application areas forest fires and floods.
- **Schema mapping**: The use case for schema mapping supports domain experts to map between local and common schemas to create harmonized forest fire information.

We identified a total of 12 use cases and 6 different systems under consideration. The figures below list all use cases, actors and systems under consideration, as well as the relationships between them.

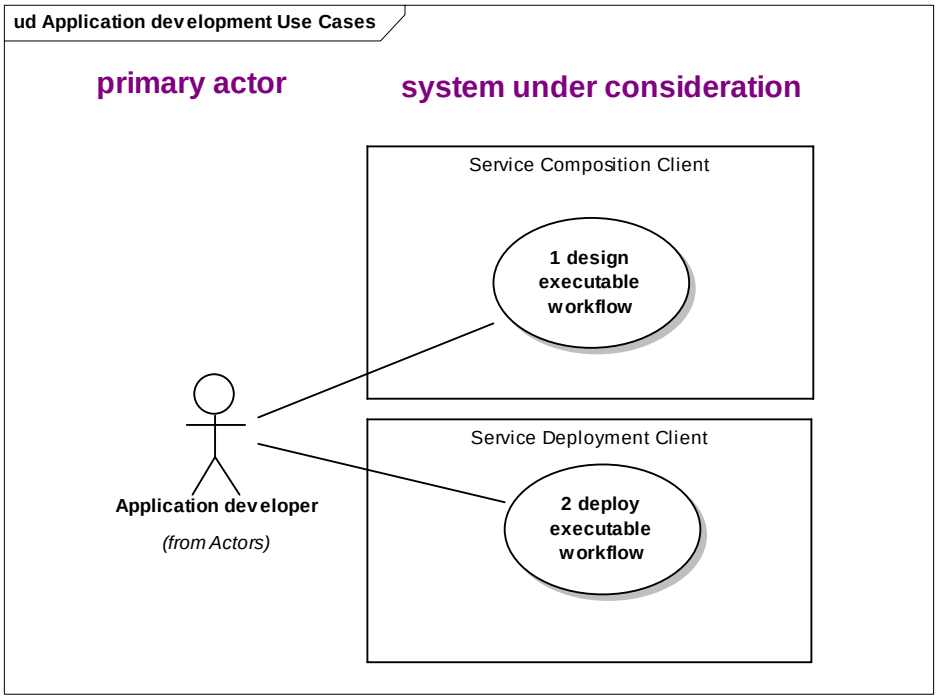


Figure 2: Use case diagram showing the identified use cases for application development.

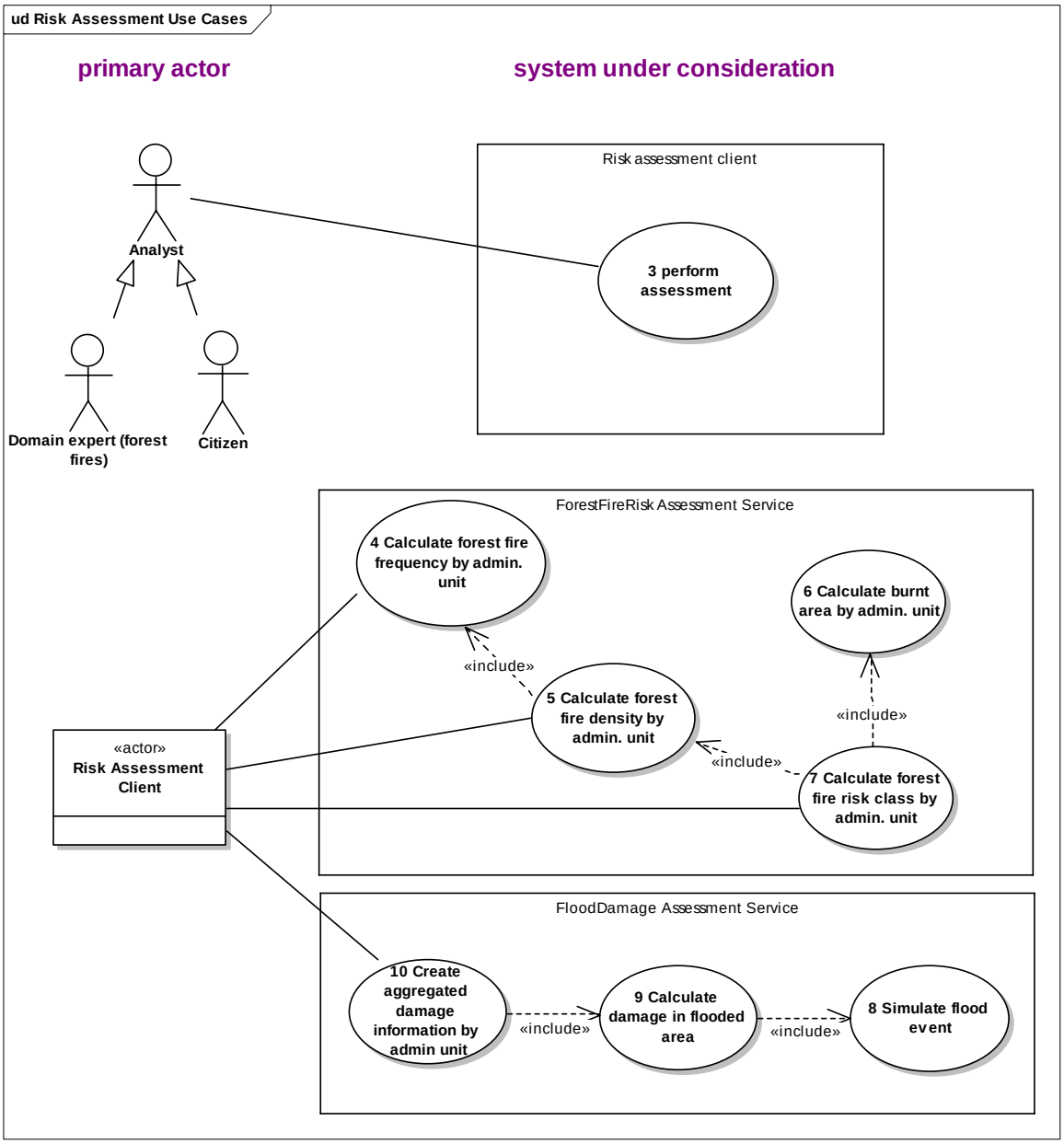


Figure 3: Use case diagram showing the relationships between the identified use cases for risk and damage assessment.

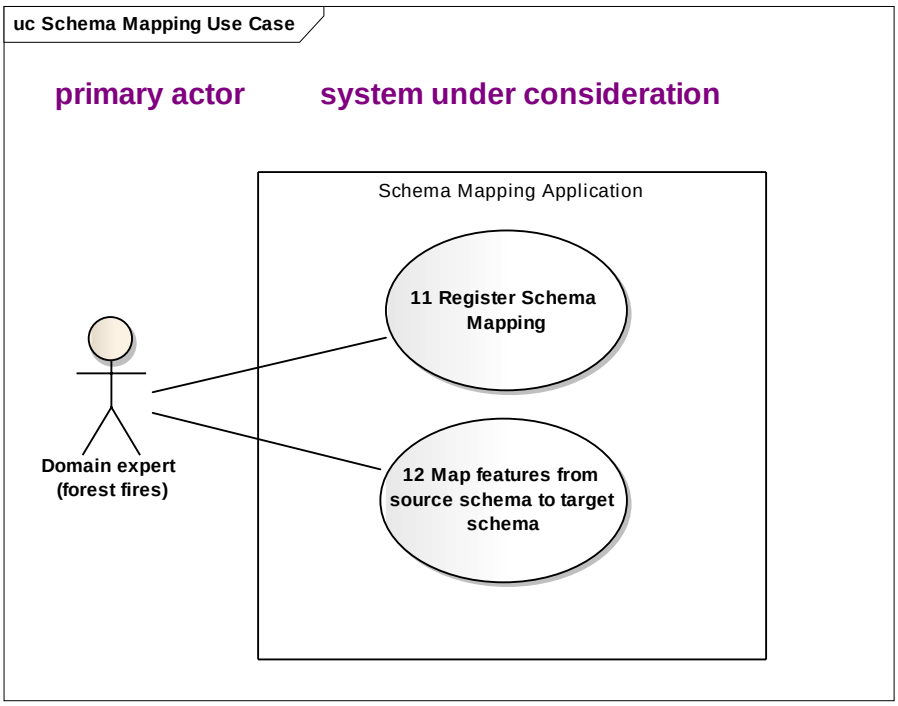


Figure 4: Use case diagram showing the use case for schema mapping.

2.2.1. Use case for development of thematic applications

In the following section the use cases are documented with a short description. Secondary actors are omitted.

1) Design executable workflow

Primary actor: developer

System under consideration: workflow design client

The developer defines, given the specific risk analysis (e.g. calculate forest fire frequency by administrative unit), the workflow of activities to be performed in order to implement the requested functionality. The workflow is encoded in a workflow descrition document.

NOTE The workflow may depend on the set of functionalities made available by the service operations. Therefore the same functionality may be achieved using different workflows. For example, the function "Calculate forest fire density by administrative unit" (c.f. section 2.2.3 Use cases in forest fire application area) has been realized by combining two existing functionalities, namely frequency and normalization, since a density function was not available as a separate operation in the service network.

2) Deploy executable workflow

Primary actor: developer

System under consideration: service deployment client

The analyst deploys the workflow description as a new service instance, which is able to execute the defined workflow on demand. The new service instance is then registered in the ORCHESTRA service network.

2.2.2. Use cases of general nature for risk assessment by analysts

3) Perform assessment

Primary actor: domain expert or citizen (= analyst)

System under consideration: risk assessment client

The analyst opens the risk assessment client and receives a graphical user interface that allows him to formulate his request. He specifies the assessment criteria, which will be the basis for a particular risk or damage assessment by interacting with the risk assessment client. This includes the area and time of interest as well as a particular assessment type and type-related parameters *(note: depending on the assessment type, use cases 4, 5, 7, or 10 are executed)*. The navigation to the area of interest is based on a selection of a bounding box by the analyst using a map display as visual aid. The analyst requests the assessment and the risk assessment client returns a map with the processing results. The visualization is defined through a symbology and accompanied by a legend. The analyst can change the symbology if required.

2.2.3. Use cases in forest fire application area

4) Calculate forest fire frequency by administrative unit

Primary actor: risk assessment client

System under consideration: risk assessment service

Based on forest fire events from European member states (points) and administrative units (polygons), the system under consideration computes the number of forest fire events by administrative area. This is done by aggregating the points based on the polygon geometry of the administrative units and using the aggregation function ‘count’. The result (fire frequency information) is linked to the administrative unit.

5) Calculate forest fire density by administrative unit

Primary actor: risk assessment client

System under consideration: risk assessment service

<<includes>> ‘Calculate forest fire frequency by administrative unit’

The system under consideration normalizes the forest fire frequency information linked to the administrative units by the area of each polygon representing an administrative unit to compute the forest fire density. Again, the result (density information) is linked to the administrative unit.

6) Calculate burnt area by administrative unit

Primary actor: risk assessment application

System under consideration: risk assessment service

Based on forest fire events from European member states (points) and administrative units (polygons), aggregate the attribute ‘burnt area’ of the forest fire events for all fires by administrative unit using the aggregation function ‘sum’. The result (‘total burnt area’) is linked to the administrative unit.

7) Calculate forest fire risk class

Primary actor: risk assessment application

System under consideration: Risk assessment service

<<includes>> ‘Calculate forest fire density by administrative area’ and ‘Calculate burnt area by administrative unit’

Based on the newly gained attributes ‘forest fire density’ and ‘total burnt area’, create a new attribute ‘forest fire class’ by using a classification rule.

2.2.4. Use cases in flooding application area

8) Simulate flood event

Primary actor: risk assessment system

System under consideration: Risk assessment service

Based on a grid coverage that shows the height difference to the closest river for each grid cell, simulate a flood by defining a flood level for the river in meter. The flood event is computed through performing a reclassification of the height differences grid coverage which is transformed into a Boolean grid coverage showing those grid cells that are flooded (flood extent).

9) Calculate damage in flooded area

Actor: risk assessment system

<<includes>> 'Simulate flood event'

System under consideration: Risk assessment service

Intersect a grid coverage showing the population density by 1000sqm with the grid coverage showing the flood extent (either being simulated in the previous use case or being a historic flood event) to create the population density exposure. The exposure is transformed into a grid coverage showing damage information by grid cell (number of people directly affected by flood). This is done by calculating the number of people living in each exposure grid cell (population density * (area of each grid cell / 1000^2)). The flood extent used for this calculation might be either simulated (see use case 8) or a historic measurement[3].

10) Aggregate damage information by administrative unit

Primary actor: risk assessment system

System under consideration: Risk assessment service

<<includes>> 'Simulate flood event' and 'Calculate damage in flooded area'

Aggregate the damage information by administrative unit (polygons). The aggregation function used is 'sum', summing up the values of all grid cells within the polygon of the administrative unit. The result (number of people directly affected by flood by administrative unit) is linked to the administrative unit polygon.

2.2.5. Use case in schema mapping application

11) Register schema mapping

Primary actor: Domain expert

System under consideration: Schema mapping application

The domain expert registers a mapping rule with the schema mapping application that describes how features in specific source schema map onto features in a specific target schema.

12) Map features from source schema to target schema

Primary actor: Domain expert

System under consideration: Schema mapping application

The domain expert requests from a schema mapping application to map (forest fire) features from a (local) source schema into a (common) target schema according to a mapping rule. The application performs the mapping and returns the features to the domain expert.

[3] The decoupling of the flood simulation and the damage assessment furthermore allows calculating the damage caused by any hazard that can be expressed through an event extent.

2.3. Functional requirements

This section contains descriptions of the functional requirements that can be derived from the use cases described above. Functional requirements describe, in simple words, *what a system is supposed to do*.

Requirement 1 'Workflow Design'. Allow the user to design complex workflows based on service instances that exist as components within an ORCHESTR Service Network. *(from Use Case 1)*

Requirement 2 'Deployment'. Allow the user to deploy the designed workflow and publish it as a new component in an ORCHESTRA Service Network. *(from Use Case 2)*

Requirement 3

3a) 'Data Access'. Provide access to a number of data sources, e.g.

- Member State forest fire registration data including the information on the location of the assumed ignition point (as geographic or projected coordinates or geographic identifier), the date and time of ignition, the burnt area (including the fraction for "forest and other wooded land area" and "non forested area") and the fire cause. All this has to be available in a harmonized schema (see 3b).
- Administrative units (NUTS[4]) and the European grid
- Population density data
- Height difference to closest river

Allow querying the spatial, the temporal and the thematic domain of both vector and coverage data sources. *(from Use Cases 1; 3-10)*

3b) 'Schema transformation': Provide means to register and perform mapping of forest fire features from a local source into a common target schema (i.e. Member State forest fire registration data). *(from Use Cases 11-12)*

Requirement 4 'Visualization'. Visualize vector and raster information on a map. Provide the user with spatial context, such as administrative boundaries, coastlines and cities. *(from Use Cases 3;8;9)*

Requirement 5 'Flood simulation'. Allow the user to simulate a flood event in a catchment area of his choice. *(from Use Case 8)*

Requirement 6 'Assessment'. Allow query-building that covers thematic (assessment criteria: forest fire frequency | forest fire density | forest fire risk | damage), spatial (bounding box), and, if applicable, temporal (time) aspects of a risk or damage assessment to be aggregated by administrative units of the user's choice. Perform the resulting query. *(from Use Cases 3;8;9)*

Requirement 7 'Symbology'. Allow the user to create and to apply a user-defined symbology. *(from Use Case 3)*

[4] NUTS stands the French *nomenclature d'unités territoriales statistiques*), representing the Nomenclature of Territorial Units for Statistics. NUTS is a geocode standard for referencing the administrative divisions of countries for statistical purposes.

3. Architecture work: Component and service specifications for the pilot application

Section 3 addresses the architecture work. Here we list service specifications that are used within this pilot application. The subsection 3.1 describes the context that is defined in the ORCHESTRA Reference Model, i.e. the service categories in which the various service specifications fall. The ORCHESTRA Services do not provide an interface to a human user but rather to a software component requesting an operation at the service interface. The provision of such user interfaces is considered to be provided by so-called Human Interaction Components (HCI).

The sections 3.2 to 3.5 provide specifications of all services and components that are used within the pilot applications. These specifications are either coming from the ORCHESTRA Reference Model, or they have been developed in the course of the pilot.

The table below provides a mapping of the functional requirements that were identified in the previous section and the resulting specifications of services and components that meet these requirements. In brackets you find the sections that contain the specifications.

Table 1: Mapping functional requirements of Service Specifications and Components

Service Specification	Requirement and how it is met
Service Chain Access Service (SCAS) (cf. section 3.2.1)	**Requirement 2**: The SCAS allows a user to design a workflow of service invocations, based on service instances in a particular service network.
Feature Access Service (FAS) (cf. section 3.2.2)	**Requirement 2**: The FAS represents a service that can be used as a component in a workflow of service invocations. It is a generic data access service that provides access to both vector and raster data.
Schema Mapping Service (SMS) (cf. section 3.2.3)	**Requirement 3b**: The Schema Mapping Service provides the functionality to map a feature collection encoded in a source schema into another feature collection encoded in a target schema. The transformation is performed according to a mapping rule that either has been registered in the service's local repository, or that is specified on-the-fly.
Translating FAS (FAS-X) (cf. section 3.2.4)	**Requirement 3a & 3b**: The FAS-X is a combination of the interfaces of the FAS and the SMS: It allows registering a mapping from a source to a target schema for a dataset accessible through a FAS. By doing so, heterogeneous forest fire data available in source schemata from member states can be directly requested from the source to be translated into a common forest fire target schema.
Map and Diagram Service (MAS) (cf. section 3.2.3)	**Requirement 4**: The MAS visualises both vector and raster data. **Requirement 7**: The MAS visualises data based on a symbology. This symbology can be either predefined or created by the user on the fly.
Aggregation Interface (cf. section 3.3.2.1)	**Requirement 2**: The Aggregation Interface represents a service that can be used as a component in a workflow of service invocations. **Requirement 6**: The interface allows to aggregate point- and polygon data, and thus to perform the function that is required for the calculation of the forest fire frequency.
Normalise Interface (cf. section 3.3.2.2)	**Requirement 2**: The Normalise Interface represents a service that can be used as a component in a workflow of service

	invocations. **Requirement 6**: The interface allows to normalise the aggregated fire frequency information by area and thus to perform the function that is required for the calculation of the forest density information.
Classification Interface (cf. section 3.3.2.3)	**Requirement 2**: The Classify Interface represents a service that can be used as a component in a workflow of service invocations. **Requirement 7**: The interface allows classifying the fire density information into risk classes.
Map Algebra Interface (cf. section 3.3.2.4)	**Requirement 2**: The Map Algebra Interface represents a service that can be used as a component in a workflow of service invocations. **Requirement 5**: The Map Algebra Interface allows the simulation of a flood event. **Requirement 6**: The Map Algebra Interface allows all the processing steps to calculate the damage assessment steps described in the use cases.
Forest fire risk assessment service (FFRAS) (cf. section 3.4.1)	**Requirement 2**: The FFRAS represents a service that can be used as a component in a workflow of service invocations. **Requirement 6**: The interface allows querying forest fire frequency, density and risk in a spatiotemporal manner through a single service interface.
Flood Simulation Service (FSS) (cf. section 3.4.2)	**Requirement 2**: The FSS represents a service that can be used as a component in a workflow of service invocations. **Requirement 5**: The interface allows simulating a flood for a given catchment based on a numeric input value describing the flood height.
Damage Assessment Service (DAS) (cf. section 3.4.3)	**Requirement 2**: The DAS represents a service that can be used as a component in a workflow of service invocations. **Requirement 6**: The interface allows querying the damage (number of people affected) that is caused by an event of a certain extent, through a single service interface.
Human interface component specification	**Requirement and how it is met**
Forest Fire Application Client & Map Viewer (cf. section 3.5.1)	**Requirement 4**: The Client provides a GUI for the user to visualise spatial context, such as administrative boundaries, coastlines and cities. **Requirement 6**: The client provides a GUI for the user to form spatiotemporal and thematic queries in order to assess forest fire density, frequency and risk. **Requirement 7**: The client provides a GUI for the user to define his own symbology, which can be applied to the displayed assessment results.
Flood Simulation & Damage Assessment Application (cf. section 3.5.2)	**Requirement 4**: The Client provides a GUI for the user to visualise spatial context, such as administrative boundaries, coastlines and cities. **Requirement 6**: The Client provides a GUI for the user to simulate flood extents and to perform damage assessment for number of people affected, based on flood extents.
Workflow Design Engine (This has not been further specified, as various software solutions for Workflow Design Engines already exist (e.g. ActiveBPEL Designer))	**Requirement 1**: The Workflow Design Engine provides a GUI for visually combining services into complex service chains.

3.1. Context defined in the ORCHESTRA Reference Model

According to the Reference Model for the ORCHESTRA Architecture (RM-OA)[5] [3], ORCHESTRA Services are functionally classified in service categories. The main service categories are ORCHESTRA Architecture Services (OA Services) and ORCHESTRA Thematic Services (OT Services):

An OA Service provides a generic, platform-neutral and *application-domain-independent* functionality.

An OT Service provides an *application-domain-specific functionality* built on top and by usage of OA Services and/or other OT services.

OT Services provide application domain-specific functionality. However, within and also between different application-domains high-level functions may be identified that have a generic nature. These services are inside the scope of the RM-OA as a generic architecture, and are defined as follows:

OT Support Service: generic service that facilitates the development or interactive composition of thematic functionality.

Taken the application domain of environmental risk management, the ORCHESTRA project provides dedicated OT Services according to the following structure:

OT Risk-neutral Service: service specific to the risk management domain that facilitates the development or interactive composition of risk-neutral risk management functionality.

OT Risk-specific Service: service specific to a specific risk management domain (e.g. earthquakes, forest fires, flood, systemic risks) that facilitates the development or interactive composition of risk-specific risk management functionality.

All OT Services may use and combine the OA Services in order to fulfil their thematic function.

HCI are software components that provide the (usually graphical) user interface (GUI) of an OA Service or OT Service (e.g. Viewers and Editors) are not described as part of the RM-OA.

[5] Some excerpts from the RM-OA have been included here as introduction to the functional classification of the ORCHESTRA services

3.2. OA and OA Support Services

3.2.1. Service Chain Access Service

Name	Service Chain Access Service
Standard Specifications	• DAML OWL-S Web Service Ontology version 1.1 (http://www.daml.org/services/owl-s/) • ESSI Web Service Modelling Ontology (WSMO) and Web Service Modelling Language (WSML) (http://www.wsmo.org) • ISO 19119:2005 Geographic information – Services • OASIS Web Services Business Process Execution Language (WSBPEL) http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel • OGC 05-007r4 Web Processing Service (WPS), version 0.4.0 (discussion paper) • OMG Business Modelling and Integration DTF (http://bmi.omg.org/) • W3C Web Service Choreography Interface (WSCI) 1.0 (http://www.w3.org/TR/wsci/) • W3C Web Services Description Language (WSDL) 1.1 (http://www.w3.org/TR/wsdl)
Description	The Service Chain Access Service supports the creation of an executable service instance based on an explicit description of a service chain. The chain can then be executed as a single service. However, the execution of the service is outside the scope of the Service Chain Access Service (see comment below). Based on the Reference Model of Open Distributed Processing (ISO/IEC 10746-1RM-ODP) definition of chain of actions, a service chain is defined in ISO 19119 as a sequence of services where, for each adjacent pair of services, occurrence of the first action is necessary for the occurrence of the second action. For the scope of this specification, it is important to distinguish between the *description of a service chain* (i.e. a document in some workflow language, e.g. BPEL), a *deployed instance of a chain* (i.e. an executable piece of code) and the actual process of *executing the chain*. The service specification is based on the aggregate service pattern where services appear as a single service which handles all coordination of the individual services that are part of the chain. The *createServiceChain* operation supports a service provider in creating an executable instance of an aggregate service based on an explicit service chain description and optionally to register that service instance with a catalogue service. The Service Chain Access Service provides the functionality through the following interfaces: • *ServiceCapabilities*: Informs the client about the common and specific capabilities of the Service Chain Access Service. • *ServiceChainAccessService:* Selection of service chain descriptions and creation and deletion of aggregate services based on such descriptions.
Interface *ServiceCapabilities (from OA Basic Service)*	

get Capabilities	Informs the requestor about the capabilities of a Service Chain Access Service. As specific capabilities, it contains the supported workflow language in which the service chain description can be specified.
Interface *ServiceChainAccessService*	
createService Chain	Deploys the service chain instance (an aggregated service) specified in a workflow document.
getService Chain	Gets a descriptor of the service chain which includes MI (id, address, description, and workflow language) and the workflow description itself.
deleteService Chain	Deletes a service chain instance.
Example usage	A client creates an aggregate service which can access features and perform schema transformations. This service can now be accessed as one single service from a client.
Comments	In a service implementation the *Service Chain Access Service* and *Processing Service* interfaces can be combined. The workflow language can then be used to define combinations of several processing operations of this service instance. Thus, a combination of related processing operations can be executed with one call without having to send the same data repeatedly to the service.

3.2.2. Feature Access Service

Name	Feature Access Service
Standard Specifications	• ISO/IEC 9075 Information technology -- Database languages -- SQL • ISO 19109:2005 Geographic information -- Rules for application schema • ISO 19125-1:2004 Geographic information -- Simple feature access -- Part 1: Common architecture • ISO 19125-2:2004 Geographic information -- Simple feature access -- Part 2: SQL option • ISO/DIS 19136 Geographic information -- Geography Markup Language (GML) • OGC 99-050 Simple Features Implementation Specification for OLE/COM V1.1 • OGC 99-054 Simple Features Implementation Specification for CORBA V1.0 • OGC 03-105r1 Geography Markup Language (GML) Encoding Specification V3.1.1 • OGC 04-094 Web Feature Service (WFS) Implementation Specification) V1.1 • OGC 04-095 Filter Encoding Implementation Specification V1.1 • OGC 05-076 Web Coverage Service (WCS) Implementation Specification (Corrigendum) V1.0.0 • OGC 05-126 Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture V1.1.0 • OGC 05-134 Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option V1.1.0
Description	The Feature Access Service (FAS) allows interoperable read and write access on feature instances available in a service network. Furthermore, the FAS provides an interface that may be inherited by more specific FAS (e.g., sensor access service) using interface inheritance. The FAS offers information about:

	• The feature types it is capable to provide. • The supported encoding(s) to transfer requested or submitted feature data. • The query language and mechanism that is supported for filtered feature access. Features provided by the FAS are instances of a certain feature type defined in an ORCHESTRA Application Schema (OAS), which again is an instantiation of an OMM_FeatureType. This means that the FAS only permits access to information which is represented through feature types according to the rules of the ORCHESTRA Meta-Model (OMM). Whether information is remodelled on-the-fly by a software component or whether the features are actually stored in a feature store is not crucial for the FAS. Seen from the interface, the feature representation is a black box and is not visible for clients. The FAS allows queries to select certain features based on their type, certain attribute values and their spatial and temporal extent. The selection statement is encoded using a query language that supports all these functionalities (e.g., SQL including spatio-temporal statements). By selecting and retrieving features, access to their attributes and operations is provided. Any FAS (and its possible profiles or possible inheriting interfaces) may support the update of existing feature instances, the creation of new feature and the deletion of existing features, and hence, in this case, it should also be transactional. It can also allow the creation, updates, and deletions of feature types. Feature instances and feature types are identifiable by a Unique Identifier (UID) that is unique with respect to at least one ORCHESTRA Service Network. If a FAS is used to create a new feature instance or feature types it will also create an appropriate UID for this feature type or instance. Additionally, it is important to emphasize the requirements for Authorisation and authentication in order to support creation, deletion, and modification of feature and feature types. The FAS provides the functionality through the following interfaces: • *ServiceCapabilities*: Informs the client about the common and specific capabilities of the Name Service. • *FeatureAccessService*: selection, creation, update and deletion of feature instances and feature types.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilities	Informs the client about the capabilities of a FAS service instance. As specific capabilities, the FAS provides information about the supported feature types, the encoding of feature type requests, the encoding of returned feature collections as well as the supported query language.
Interface *FeatureAccessService*	
getFeature Types	Gets a description (the schema) of given feature types serviced by an FAS instance in a specific encoding based on a query.
setFeature Types	Updates existing Feature Types matching a given query.
createFeature Types	Creates new Feature Types based on feature type descriptions.
deleteFeature Types	Deletes existing Feature Types matching a given query.
getFeatures	Retrieves features and their attributes matching a given query.
setFeatures	Updates existing features matching a given query.
createFeatures	Creates new features based on a feature collection and a given query.
deleteFeatures	Deletes existing features matching a given query.

Example usage	A client accessing this service wants to retrieve all feature instances of roads for a particular region. The Feature Access Service is passed a getFeatures request for the specified area and feature type. A response is generated containing all valid features. The features may be modified and submitted to the Feature Access Service as an update transaction (via the setFeatures operation).
Comments	As the RM-OA, in accordance with ISO 19123, considers coverages as subtypes of features, the FAS can also be used to access coverages.

3.2.3. Schema Mapping Service

Name	Schema Mapping Service
Standard Specifications	No standard service specification currently exists, on which the functionality of the Schema Mapping Service could be based. Several standards exist, on which a language for describing a schema mapping can be based. However, as the Schema Mapping Service does not define a specific schema mapping language, it is up to an implementation specification to define these. Prominent (draft) standards which can be used for describing a schema mapping are: • W3C XSL Transformations (XSLT), version 1.0 (http://www.w3.org/TR/xslt/) • XQuery 1.0: An XML Query Language, W3C Recommendation (http://www.w3.org/XML/Query/) • W3C SPARQL Query Language for RDF, W3C Working Draft, 4 Oct 2006 (http://www.w3.org/TR/rdf-sparql-query/)
Description	The Schema Mapping Service provides functionality that is related to the mapping of features from a source into a target schema. It provides this functionality through two interfaces. The main functionality of the SchemaMapping interface is to execute a schema mapping. A schema mapping is considered to be “the definition of an automated transformation of each instance of a data structure A into an instance of a data structure B that preserves the intended meaning of the original information”. The service takes a feature collection and a description of the mapping from the source to the target schema as input and returns the features in the target schema. A schema mapping is described by • an identifier that is unique to the Schema Mapping Service instance; • descriptions of the source and target feature types; • the schema mapping language used to describe the mapping; and • a reference to the actual mapping. The Schema Mapping Service can be used to (1) directly map from one application schema to another one, or (2) to map from an application schema to a common (or community) schema (or vice versa). The latter can be used to perform an indirect mapping between two application schemas through the community schema. The mapping of features might also require that several feature collections be combined. In order to support this, an optional concatenation operation is also included in the interface. The description of the schema mapping is required as an input. It is outside the scope of the Schema Mapping Service to automatically derive a mapping between two application schemas.

	The SchemaMappingRepository interface supports repository functionality for mappings between source and target feature types. Service can also serve as a repository for mappings between source and target feature types. For this, operations for the creation (registration), retrieval, updating and deletion of schema mapping descriptions are foreseen. The Schema Mapping Service provides its functionality through the following interfaces: • ServiceCapabilities: Informs about the common and specific capabilities. • SchemaMapping: Execution of schema mappings and concatenation of feature collections. • SchemaMappingRepository: Creation, deletion, update and selection of schema mappings.
Interface *ServiceCapabilities*	
getCapabilities	Informs the requestor about the common and specific capabilities of a Schema Mapping Service instance. Examples of specific capabilities are the supported schema mapping language (for the Schema Mapping interface) and a list of the mappings registered with the service (for the Schema Mapping Repository interface).
Interface *SchemaMapping*	
mapFeatures	Maps a feature collection to a target schema.
concat	Concatenates several feature collections.
Interface *SchemaMappingRepository*	
createMapping	Registers a new mapping with this instance of the Schema Mapping Service.
getMapping	Returns a (list of) mapping(s) matching a given query.
setMapping	Updates a specific mapping.
deleteMapping	Deletes all mapping matching a given query.
Example usage	A client wants to transform a data source in a local schema into a common agreed global schema. The client submits a feature collection and mapping rules specifying how to map the features into the required feature type.
Comments	The described interfaces can be used in service implementations in different ways: • A service that only implements the *SchemaMapping* interface can be used to map feature collections in arbitrary schemas to a target schema using a mapping description that is provided by the requester. • A service that implements both interfaces can be used in the same way. In this scenario, the requester does not necessarily have to provide the mapping description themselves but can query the Schema Mapping Service for an appropriate mapping description. • A service that implements the *SchemaMappingRepository* interface and another interface for creating or accessing feature collections (e.g. the interfaces of the Feature Access Service or the Processing Service) can be used to provide the output feature collections in different schemas.

3.2.4. Translating Feature Access Service

Name	Translating Feature Access Service (FAS-X)
Standard Specifications	*Please refer to the corresponding sections of the Feature Access Service and the Schema Mapping Service*

Description	The translating Feature Access is a combination of several interfaces. The principles of The FAS-X provides the functionality through the following interfaces: • Feature Access Service • Schema Mapping Interface
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilities	Informs the client about the capabilities of an FAS-X OSI. As specific capabilities, the FAS-X provides information about the supported feature types, the encoding of feature type requests, the encoding of returned feature collections, the supported query languages and the registered mappings.
Interface *FeatureAccessService*	
getFeatures	Retrieves features and their attributes matching a given query.
getFeature Types	Gets a description (the schema) of given feature types serviced by an FAS-X instance in a specific encoding based on a query.
Interface SchemaMapping of the Schema Mapping *Service*	
createMapping	Creates and registers a new mapping with this instance of the Translating Feature Access Service.
getMapping	Returns a (list of) mapping(s) matching a given query.
deleteMapping	Deletes all mapping matching a given query.
setMapping	Not considered in this implementation specification.
Example usage	A client accessing this service wants to retrieve all feature instances of forest fires in a common schema that has previously been registered through the createMapping operation.
Comments	The interface of the FAS-X is a combination of the Feature Access Service and the Schema Mapping interfaces.

3.2.5. Map and Diagram Service

Name	Map and Diagram Service
Standard Specifications	• ISO 19128:2005 - Geographic information -- Web map server interface • ISO/DIS 19136 Geographic information -- Geography Markup Language (GML) • OGC 02-070 Styled Layer Descriptor (SLD) Implementation Specification V1.0 • OGC 04-094 Web Feature Service (WFS) Implementation Specification) V1.1 • OGC 04-095 Filter Encoding Implementation Specification V1.1 • OGC 05-076 Web Coverage Service (WCS) Implementation Specification (Corrigendum) V1.0.0 • OGC 06-042 Web Map Service (WMS) Implementation Specification V1.3.0
Description	The Map and Diagram Service is a service that visualizes, symbolizes and enables geographic clients to interactively visualise geographic and statistical data. Its main task is to transforms geographic data (vector or raster) and/or numerical tabular data (e.g. census data, result of a statistical analysis) into a graphical representation using symbolization rules.

	The main output of this service is an image document, which can be either in raster (e.g. jpeg, png) or symbolized-vector format (e.g. SVG). The meaning of the image document (the output of this service) is a general reference map (visualization of geographic information), a diagram (visualization of statistical data) or a thematic map (visualization of the spatial distribution of one or more statistical data themes). This service enables the integration of extended Style Layer Descriptor (SLD) documents, which allows the definition of symbologies and symbolization rules at the feature level and allows also the integration of user data and remotely available data from other OA Services like the Feature Access Service. The Map and Diagram Service provides the functionality through the following interfaces: • *ServiceCapabilities*: Informs the client about the common and specific capabilities of the Map and Diagram Service. • *MapDiagramService*: This interface allows a client to request and receive maps, diagrams and, optionally, information about the visualized features according to specifications, as well as to put/remove data and symbologies on the server for visualization.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilities	Informs the client about the capabilities of a Map and Diagram Service OSI. As specific capabilities, the Map and Diagram Service instance provides a document containing, among others, a list of supported operations and predefined data layers available on the server with the corresponding layer information.
Interface *MapService*	
getMap	Returns a map of spatially referenced geographic and thematic information as an image document with the characteristics specified by the client application. The characteristics of the output image are specified by the *outputAttributes* parameter (image format, width, height, transparency, etc…) as well as the *mapAttributes* parameter (list of layers and their corresponding styles, coordinate reference system, global bounding box). Optionally, the map parameters can be provided using an SLD document.
getDiagram (optional)	Returns a diagram representation of numerical data as an image document with the characteristics specified by the client application. The characteristics of the output image are specified by the *outputAttributes* parameter (image format, width, height, transparency, etc…) as well as the *diagramAttributes* parameter (list of tabular data layers and their corresponding styles – diagram type, diagram characteristics). Optionally, the diagram parameters can be provided using an SLD document. This operation expects that the data to be rendered is in tabular format.
getLayer Legend (optional)	Returns a legend symbol (corresponding to a layer) as an image document with the characteristics specified by the client application. The characteristics of the output image are specified by the *outputAttributes* parameter (image format, width, height, transparency, etc…) as well as the *styledLayer* parameter (name of the layer for which the legend should be generated and its corresponding styles). If the styles corresponding to the layer are not available on the server, then the styles have to be defined and sent again by the client (optionally, also as a SLD document).
getFeatureInfo (optional)	Returns information about the features rendered in a certain point of a map or diagram layer as a document. The request must specify the attributes of the query point (x and y coordinates of the point in the image coordinate system, the layer name, and the number of features for which is expected to receive information) as well as a copy of the request that generated the image.
putLayer (optional)	Stores a new data layer on the server if the format of the sent layer data is supported (the supported formats for data input are advertised in the service capabilities). For this operation the following information must be defined: the layer (name, data, data format, minimum and maximum scale, etc…), the duration for which the layer will be stored and also if it will be visible or not for other users. The operation confirms the success of the request by sending back to the client a Boolean “TRUE”.
removeLayer	Removes an existing data layer from the server. The operation confirms the

(optional)	success of the request by sending back to the client a Boolean “TRUE”.
putStyle (optional)	Stores a new style layer on the server. For this operation the style must be defined either by sending the symbology or by referencing a remotely available symbology. Furthermore, the duration for which the style will be stored and also if it will be visible or not for other users must be defined. The operation confirms the success of the request by sending back to the client a Boolean “TRUE”.
removeStyle (optional)	Removes an existing style from the server. The operation confirms the success of the request by sending back to the client a Boolean “TRUE”.
describeLayer (optional)	Returns a layer description document containing schema information for a layer: attribute names, types, units, and statistical information (when applicable) like value ranges, max, min etc.). This information is needed by clients in order to create their own styles and symbolization rules based on attribute values.
Example usage	A requestor accessing this service wants to create a map that shows the spatial distribution of the hazard zones (classified by the susceptibility level) with different colours. On top of this layer the requestor is interested to have the road network, the hydrological network, and the urban areas. The hazard zones data and the data for the hydrological network are accessible by means of a Feature Access Service and other layers are available on the server. The requestor now invokes a *getMap* operation by passing a styled layer descriptor document, which defines the location of the data and the symbolization of each layer. The response of the service will be a map provided in the requested format.
Comments	It is beyond of the scope of this service to provide a human interface like the geographic viewer in the human interaction services. On the other side, other map service instances, a geographic viewer or even a Web browser could act as a client to this service.

3.3. OT Risk-neutral services and operations

3.3.1. Processing Service

Name	Processing Service
Standard Specifications	• OGC 05-007r4 Web Processing Service (WPS)
Description	The Processing Service describes a common interface for services offering processing operations on spatial (vector as well as raster) and non-spatial data. Examples of processing operations are statistical or geospatial calculations, image processing and analysis or, in general, computer algebra operations. The Processing Service provides mechanisms to identify the data required by the calculation, initiate the calculation, and manage the output so that it can be accessed by the client. The Processing Service provides the functionality through the following interface: • *ServiceCapabilities*: Informs the client about the common and specific capabilities of the Processing Service. • *ProcessingService*: provides the means to get information on and to invoke a specific processing operation.
Interface *ServiceCapabilities (from OA Basic Service)*	
Get Capabilities	Informs the client about the capabilities of a Processing Service instance. As specific capabilities, the Processing Service instance provides a list of the supported processing operations (name and abstract).
Interface *ProcessingService*	
getProcess	This operation allows a client to request and receive back detailed information about one or more processing operation(s) that can be executed by an execute

Description	operation, including the input parameters and formats, and the outputs.
Execute	This operation allows a client to execute a specified processing operation implemented by the Processing Service, using provided input parameter values and returning the outputs produced.
Example usage	A client wants to create a buffer zone around a forest during a fire and calculate the total area that is included in the buffer. The client queries the Processing Service for a description of the buffer processing operation (including its input and output types) using the *getProcessDescription* operation and then calls the buffer processing operation using the execute operation. The Processing Service returns the result of the buffer processing operation either directly or as a reference (that can be used by the client to access the result).
Comments	In order to avoid that the same data repeatedly has to be sent to the same instance of a processing service to execute several related operations, it should be possible to invoke a combination of related processing operations with one call to the service. This can be achieved by a service instance that implements both the Processing Service and the Service Chain Access Service (SCAS) interface. Thus, a SCAS workflow language can be used to define combinations of processing operations. The optimisation of "local" operation calls is an issue that should be addressed at the implementation level. For a common understanding of processing operations, (basic) operations should be grouped and described in an operation taxonomy to be referenced in the service specific capabilities. Guidelines could be e.g. the Map Algebra operations (Tomlin 1990) or the Egenhofer Operators (Egenhofer 1989).

3.3.2. Conceptual Model of the Processing Operations

This section describes a *conceptualisation* of the processing operations that will be offered by (and wrapped through) a Processing Service (PS). They are split in the two thematic application areas: forest fires and floods. The operations are grouped into interfaces. Several interfaces together can be grouped into a single Processing Service Instance.

3.3.2.1. Aggregation interface

The aggregation interface offers two operations (Figure 5):

- **LeftOuterJoin**: performs a left outer join on two feature collections using a join expression. A join expression consists of a join predicate (e.g. equal, inside) and one join attribute from each of the two feature collections. The supported join predicates should be specified in the capabilities of the service.
- **PointInPolygonAggregation**: first performs a left outer join on two feature collections (one being a polygon collection, one being a point collection), applies the "inside" join predicate, and then aggregates chosen attributes. Then aggregates attribute values of an input feature collection using one or more aggregation expressions and a (possibly empty) set of attributes to group by. An aggregation expression consists of an attribute and an aggregation function (e.g. count, mean, max, min or sum). The supported aggregation functions should be specified in the capabilities of the service.

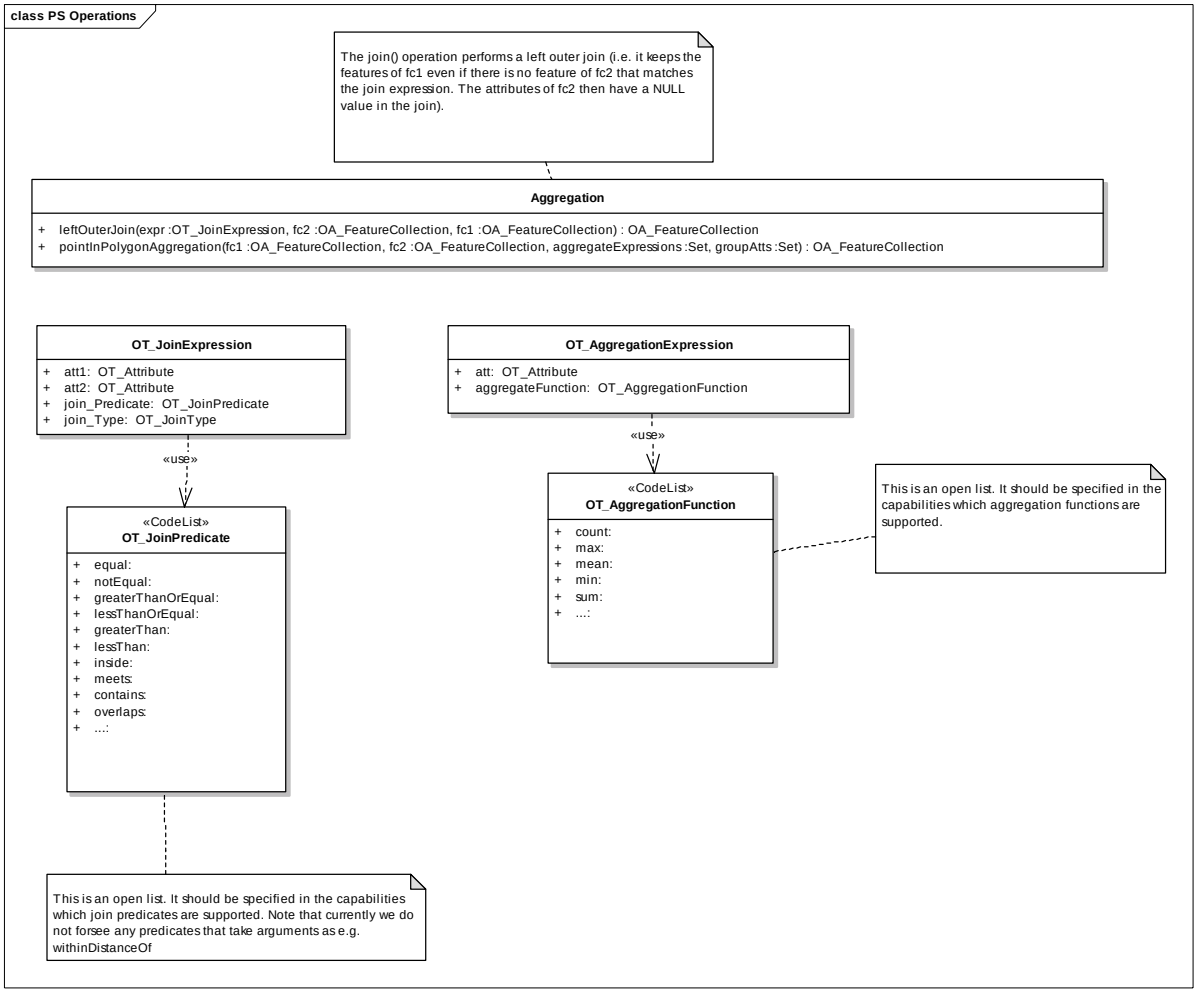


Figure 5: Conceptualisation of the processing operations offered by the aggregation interface

3.3.2.2. Normalisation interface

The normalisation interface offers one operation (Figure 6):

- **normalise**: normalises a set of attributes from the given feature collection using an attribute to normalise by. If one of the attributes is a geometry attribute, the area (for polygons) or length (for line strings) is used for the normalisation. The unit of the normalised information depends on the unit of the used Coordinate Reference System (CRS) (e.g. if the CRS uses meter and the information to normalise is a count value, the unit of the information resulting from a normalisation by area is $1/m^2$). Note that spatial normalisation can lead to wrong results if the calculation of area or distance is based on a CRS not suited for this.

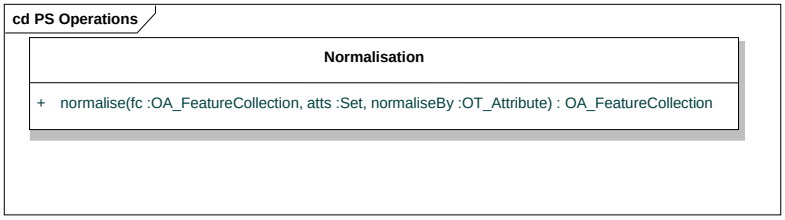


Figure 6: Conceptualisation of the processing operations offered by the normalisation interface

3.3.2.3. Classification interface

The classification interface offers 3 operations (Figure 7):

- **classify**: calculates the class ranges for an input feature collection according to the values contained in a specified field. Class ranges are calculated according to a classification method (e.g. quantile, natural breaks, equal intervals) and the requested number of classes. This operation is available only for numeric fields.
- **assignClasses**: assigns the features belonging to the input feature collection to different classes using one or more assignment clauses. Each assignment clause contains a condition (bool_expr) and the value to be assigned if the condition is true. The conditions are mutually exclusive.
- **classifyAndAssign**: merges the functionalities of the "classify" and "assignClasses" operations; the class ranges are calculated using the same statistical method as in the classify operation; the service assigns the value specified by the user to each calculated break; the input "values" parameter is assigned in the following way: i-th range created ⇔ i-th value of the parameter.

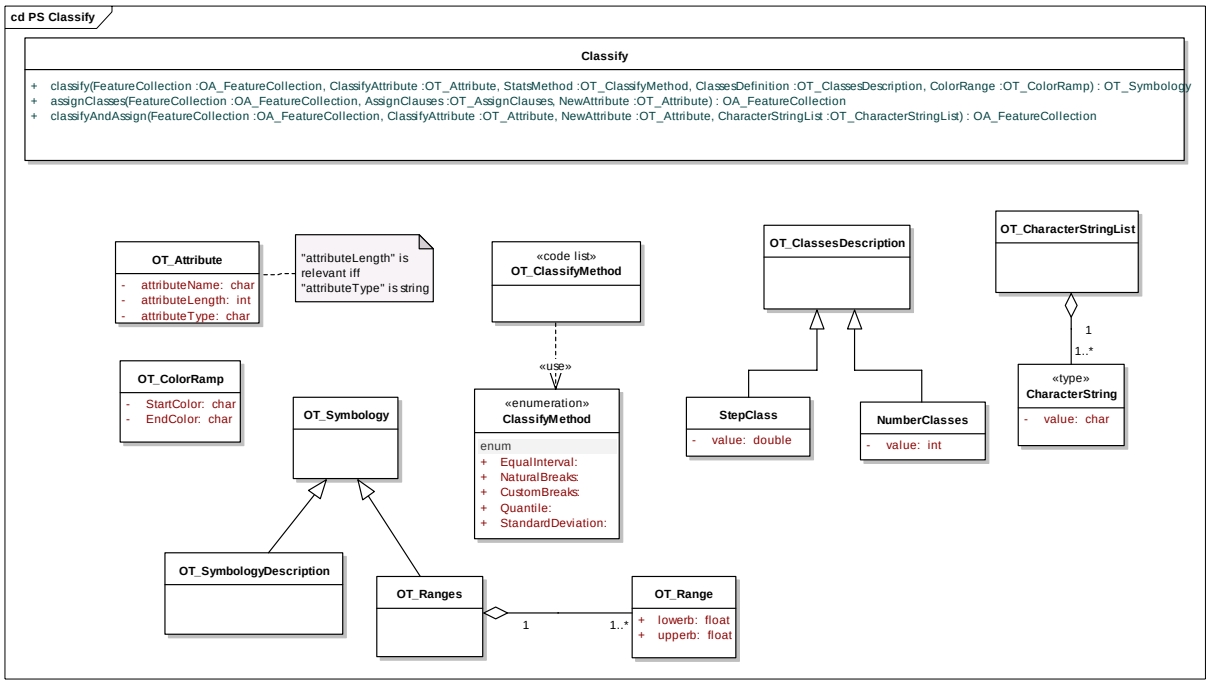


Figure 7: Conceptualisation of the processing operations offered by the classification interface

3.3.2.4. Map Algebra interface

The map algebra interface offers 'local' and 'zonal' operations that are performed on coverage features [10]. The local operations cover the arithmetic binary operators and a reclassify operation based on local values. The zonal operations cover a spatial aggregation function and a reclassification, both based on a zone feature. The map algebra interface offers 7 operations (Figure 8):

Local operations

Arithmetic binary operation

- *LocalRatio*: The operation takes one input coverage (1st operand) and one number (2nd operand). It performs an arithmetic operation (Ratio) and produces one output coverage feature.

Reclassifier

- *LocalRating*: Assigns new values to locations of the input coverage. The assignment of values is based on value-ranges of the input coverage feature and a corresponding new value for the output coverage feature.

Zonal operations

Each zonal operation requires one input coverage and one zone coverage and produces one output coverage. Each of the presented zonal operations is a binary operation. Accordingly a common domain defines the spatial resolution and the extent of the result coverage. Both inputs have to match this domain. This might require resampling using a given interpolation method.

- *ZonalSum*: Sums up the values of the input coverage within each zone of the zone coverage.
- *ZonalRating*: Reclassifies an input layer based on a zone layer. New values are assigned per zone. This can be either a mapping to a constant value, or to the values of a particular coverage feature within this zone.

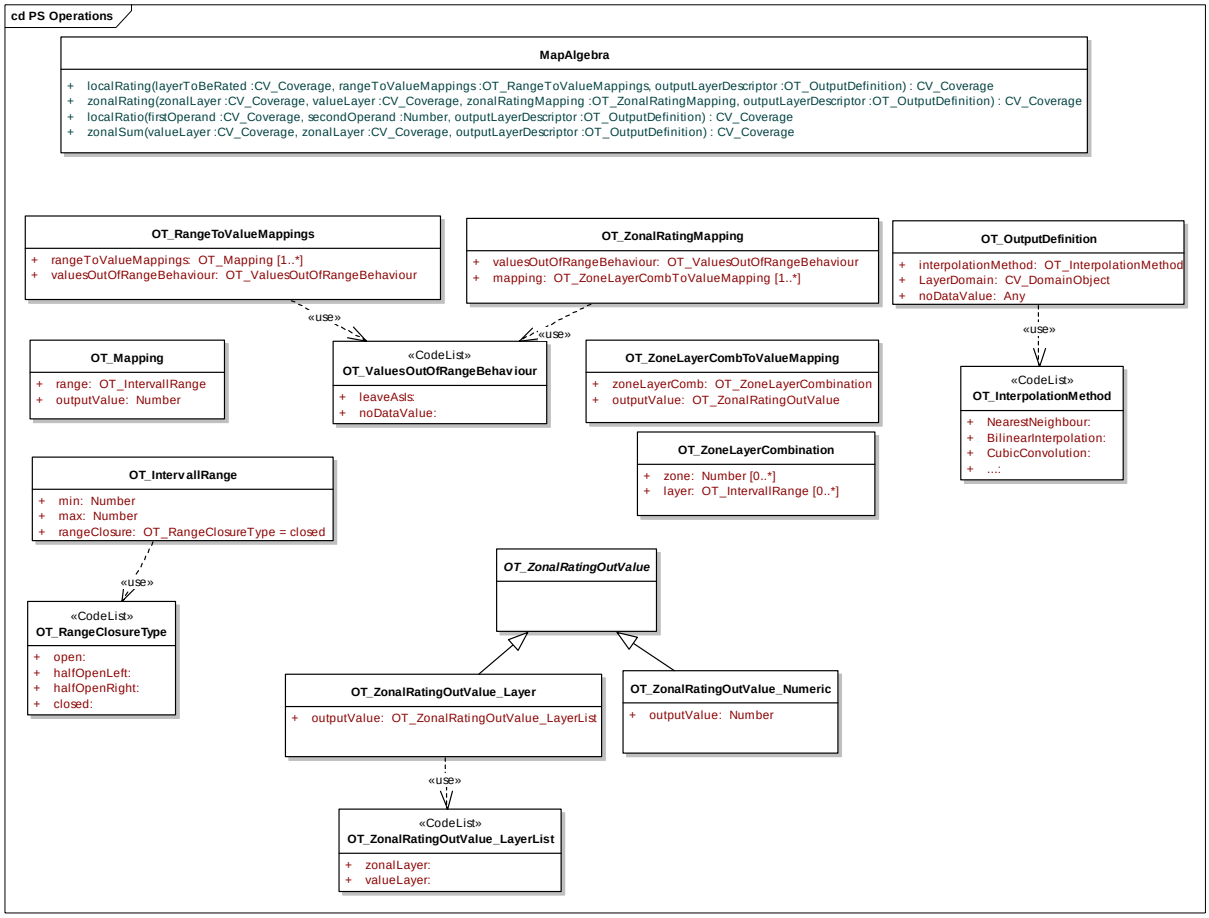


Figure 8: Conceptualisation of the processing operations offered by the Map Algebra interface

3.3.3. Mapping the Conceptual Model to Operations of the Processing Service

In order to make the processing operations described in the previous sections available through the Processing Service interface it is necessary to map the conceptual to a logical model. We already described in section 3.3.1 that the PS interface contains the three operations: *GetCapabilities*, *GetProcessDescription* and *Execute*. Each process is thus described as a profile for an execute operation, with zero or more inputs and zero or one outputs (cf. Figure 9).

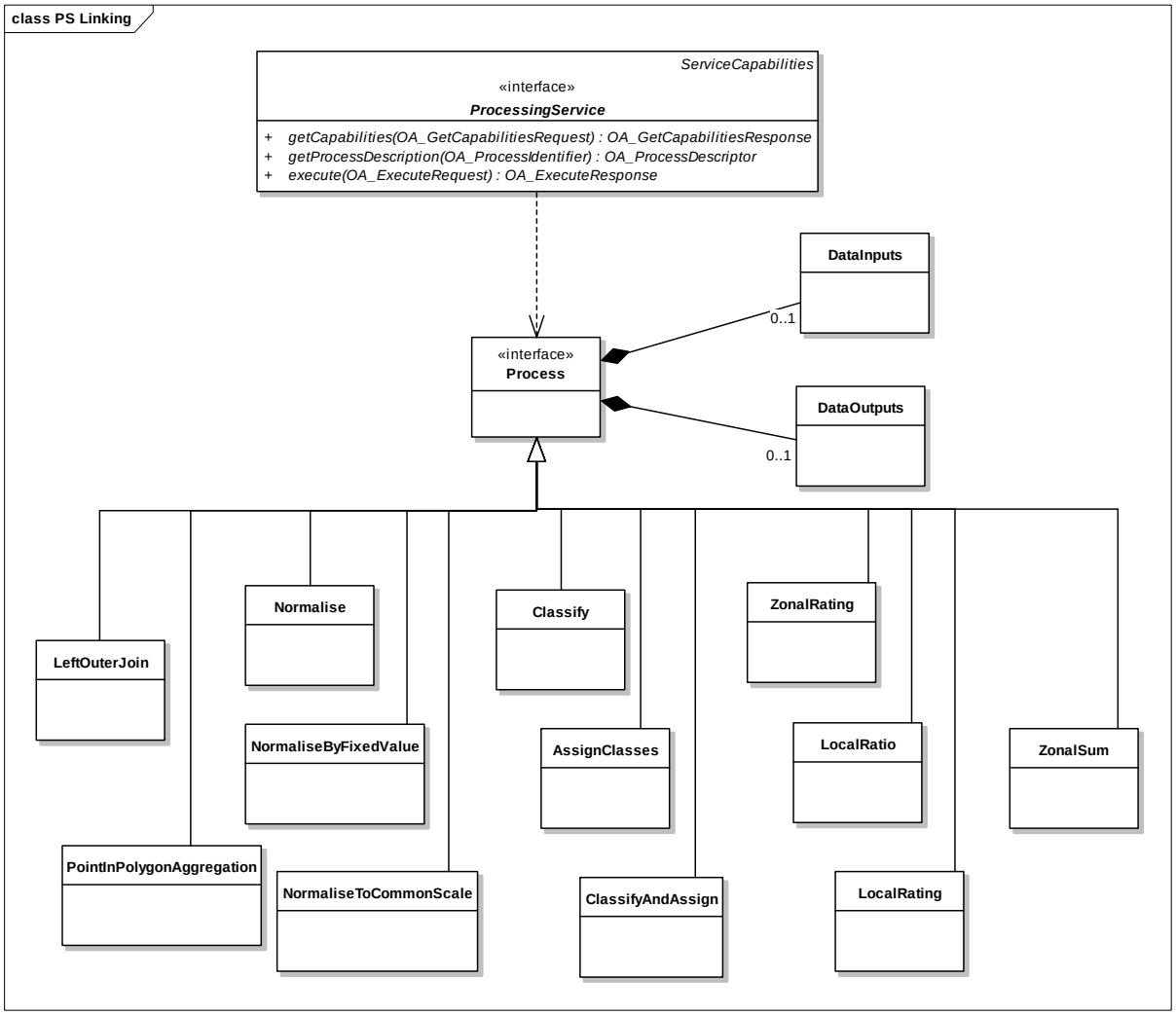


Figure 9: UML class diagram illustrating the connection between the PS interface and the particular processes

3.3.3.1. Join and Aggregation interface

Figure 10 illustrates the "LeftOuterJoin" process in detail. *Data inputs* for this process are two OA_FeatureCollections ("fc1" and "fc2") and one OT_JoinExpression ("expr"). The OT_JoinExpression consists of two attributes ("att1" and "att2") and one OT_JoinPredicate ("joinPredicate") which can be chosen from a given list. The data output of the process is a single Feature Collection that contains the joined features.

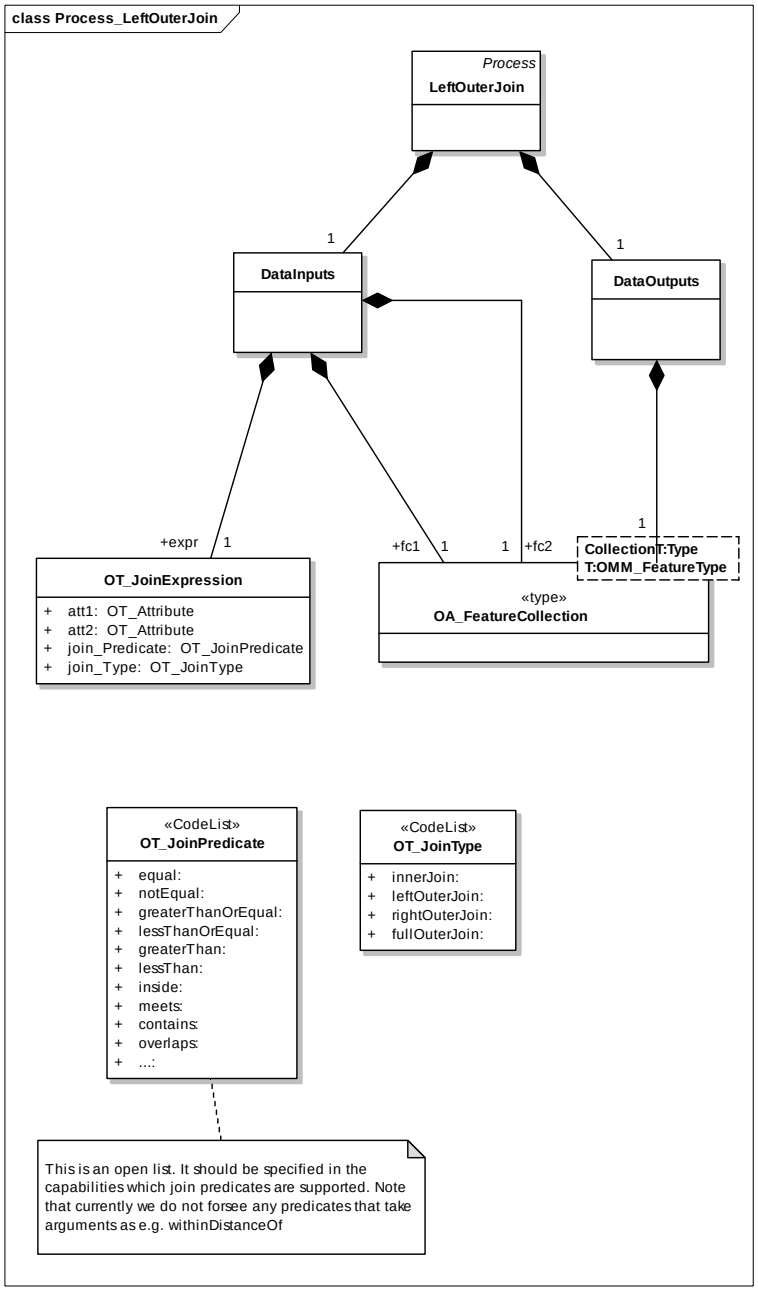


Figure 10: UML diagram illustrating the *leftOuterJoin* Process

The "PointInPolygon" process (Figure 11) has again two OA_FeatureCollections ("fc1" and "fc2") as *Data inputs* where the first one is containing point geometries and the second one containing polygon geometries. The "inside" join predicate is automatically applied the geometry attributes. Additionally, for the aggregation, it expects aggregate expressions ("aggregateExpressions") and attributes to group by ("groupAtts"). The data output of the process is a single Feature Collection that contains the joined and aggregated features.

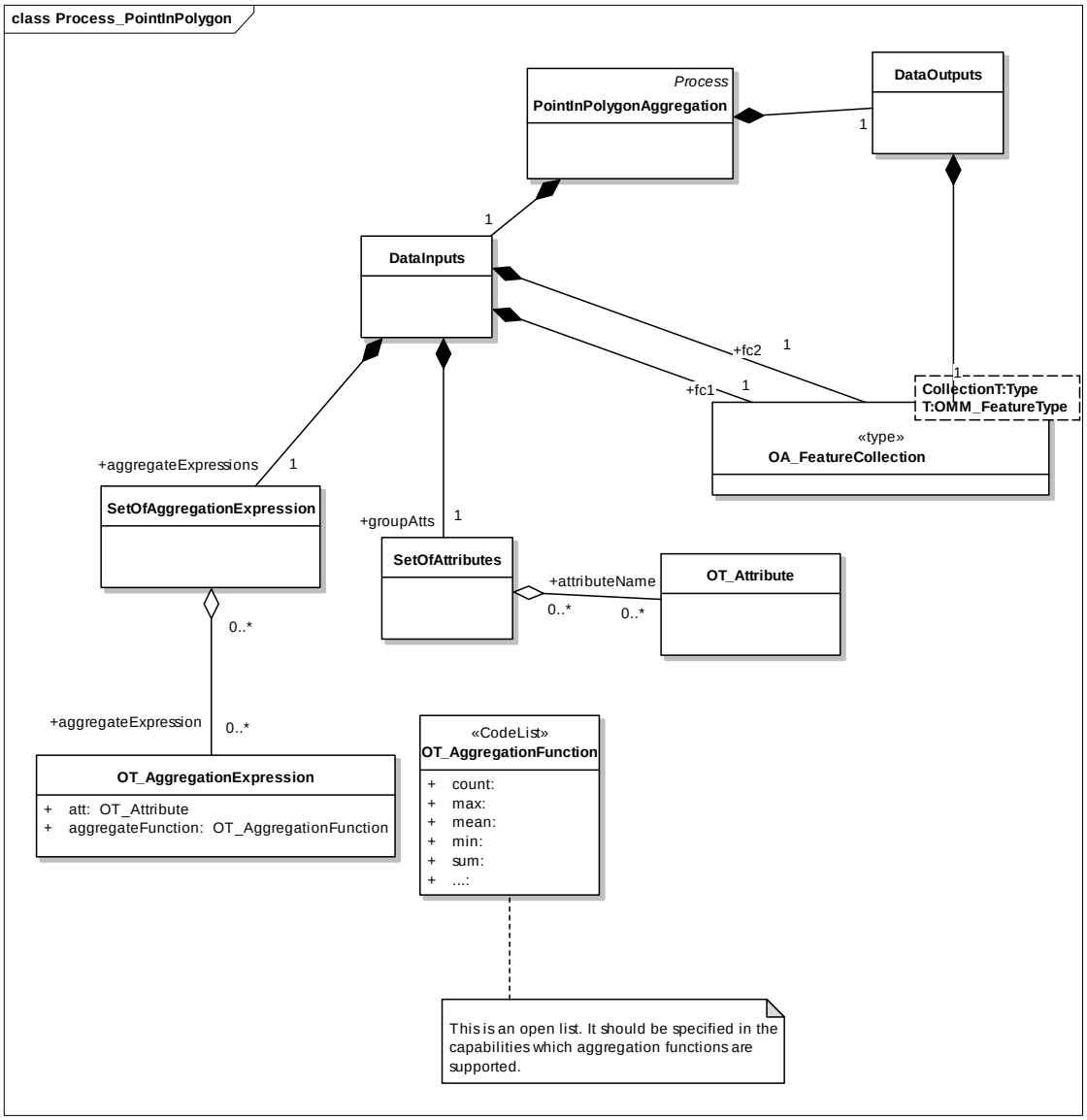


Figure 11: UML diagram illustrating the *PointInPolygon* Process

3.3.3.2. Normalisation interface

In Figure 12 the "Normalise" process is described in detail. *Data inputs* for this process are one OA_FeatureCollection ("fc1"), a set of attributes of "fc1" that shall be normalised ("atts"), and a single attribute which all members of "atts" shall be normalised by ("normaliseBy"). The data output of the process is a single Feature Collection that contains all information given in the input dataset and additionally the normalised attribute(s).

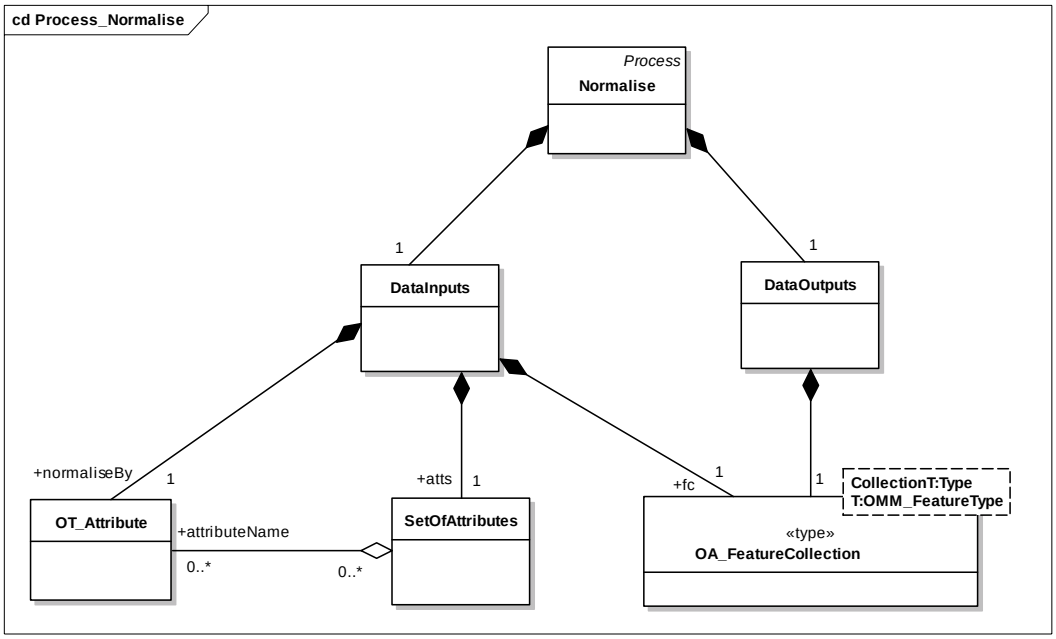


Figure 12: UML diagram illustrating the *Normalise* Process

3.3.3.3. Classification interface

Figure 13, Figure 14, and Figure 15 show in detail the operations belonging to the "classification" interface.

In Figure 13 the “Classify” process is described. The process requires four inputs: a Feature Collection (OA_FeatureCollection), the attribute of the feature which is to be classified (OT_Attribute), the chosen method to carry out the classification (OT_ClassifyMethod), and the number of classes wanted (int). The output of the “Classify” process is the list of ranges (classes) created.

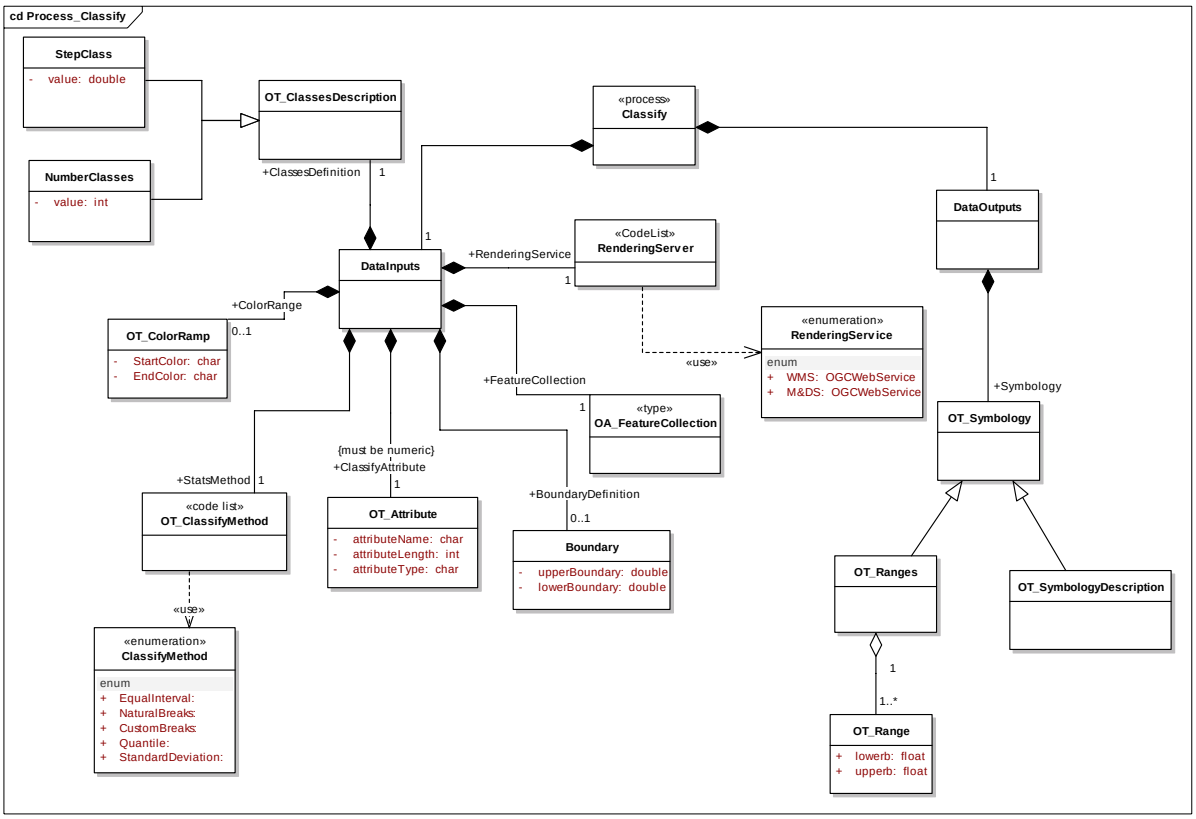


Figure 13: UML diagram illustrating the *Classify* Process

Analogously, Figure 14 shows the "AssignClasses" process in detail: the input data are given by a Feature Collection (OA_FeatureCollection), the list of conditioned expressions (OT_AssignClauses) and the specification of the new attribute which is being created (OT_Attribute). The output is represented by the updated Feature Collection. Note that the conditioned expressions are represented as couples, whose first element is a Boolean expression and the second element is the value corresponding to that expression that is the value to be assigned if the expression is true.

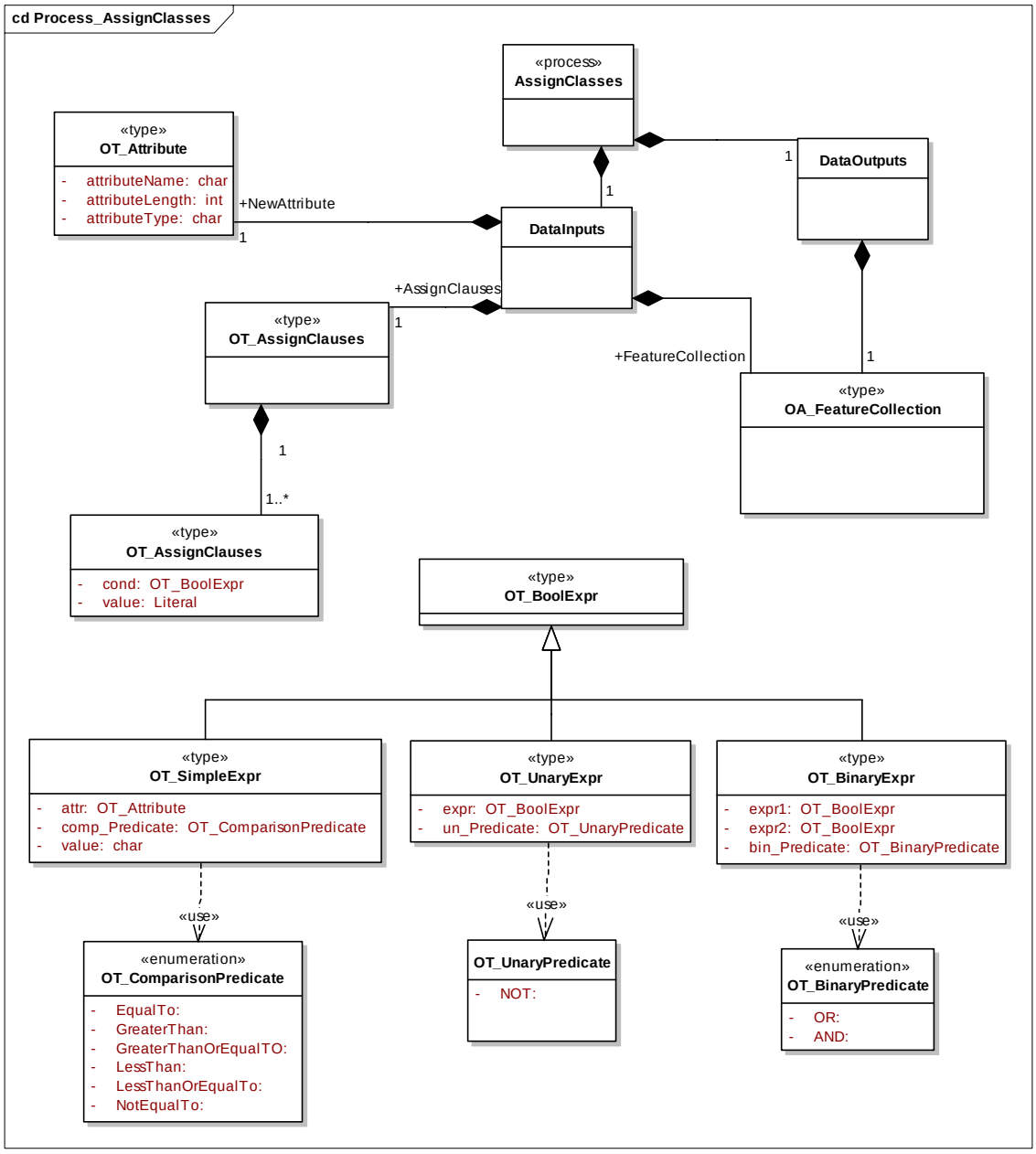


Figure 14: UML diagram illustrating the *AssignClasses* Process

Figure 15 shows the "ClassifyAndAssign" process. The process requires four inputs: a Feature Collection (OA_FeatureCollection), the attribute of the feature which is to be classified (OT_Attribute), the chosen method to carry out the classification (OT_ClassifyMethod) and the number of classes (int) wanted as in the "Classify" process. In addition, the process takes the list of values to be assigned to the ranges internally created and the specification of the new attribute (OT_Attribute) which is being created. The output is represented by the updated Feature Collection.

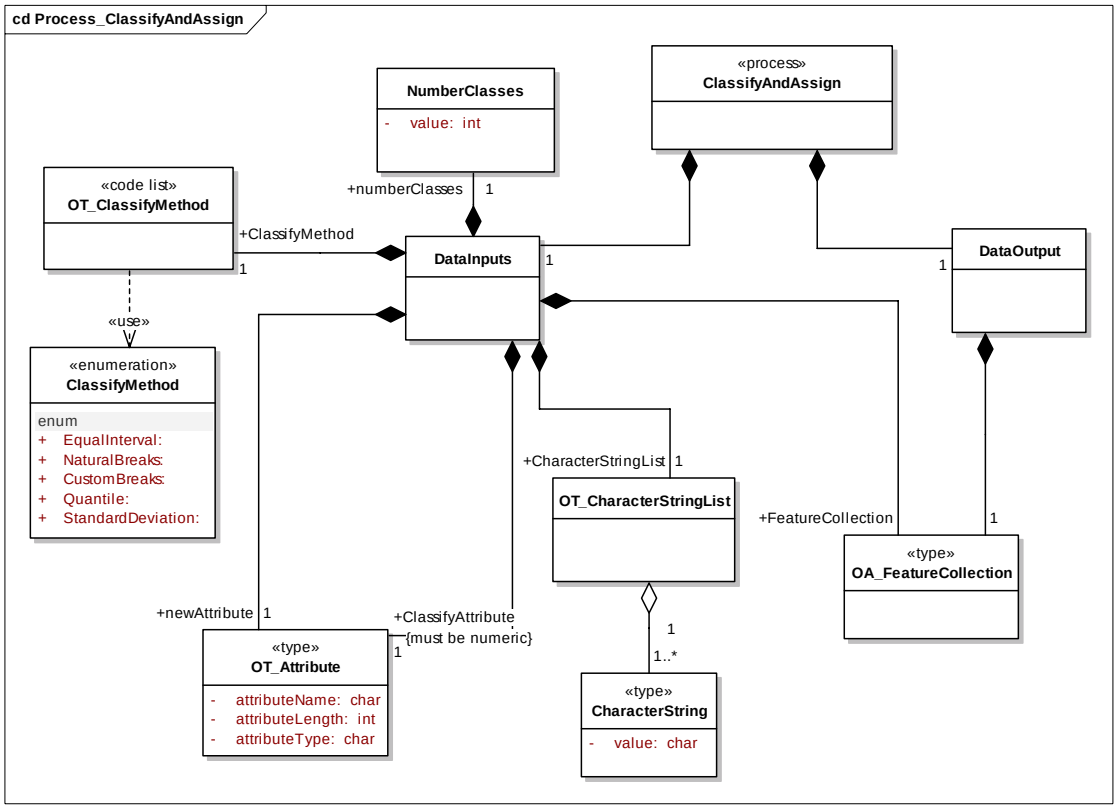


Figure 15 UML diagram illustrating the *ClassifyAndAssign* Process

3.3.3.4. Map algebra interface

Figure 16, Figure 17, Figure 18 and Figure 19 show in detail the operations belonging to the "map algebra" interface.

Figure 16 shows the "LocalRating" process: It reclassifies an input Coverage according to a particular mapping given as an input parameter and produces an output Coverage. The process takes the following inputs:

- The input layer that shall be reclassified (layerToBeRated);
- The value-mapping rules map value-ranges of the input layer to new values in the output layer (rangeToValueMappings) by allow the definition of a mapping of one to many value ranges to a single output value.
- Values that are not defined in the mapping are handled by the valuesOutOfRangeBehaviour attribute, which can be used to set all values that are not described in the mapping either a) to a no-data value or b) to keep the value of the original layerToBeRated.
- The definition of the output that determines the output Domain, the no-data value and the interpolation method that shall be used to create the output domain (if it differs from the domain of the input layer) (outputLayerDescriptor).

The output of the “Local Rating” process is:

- A Coverage that contains the result of the processing.

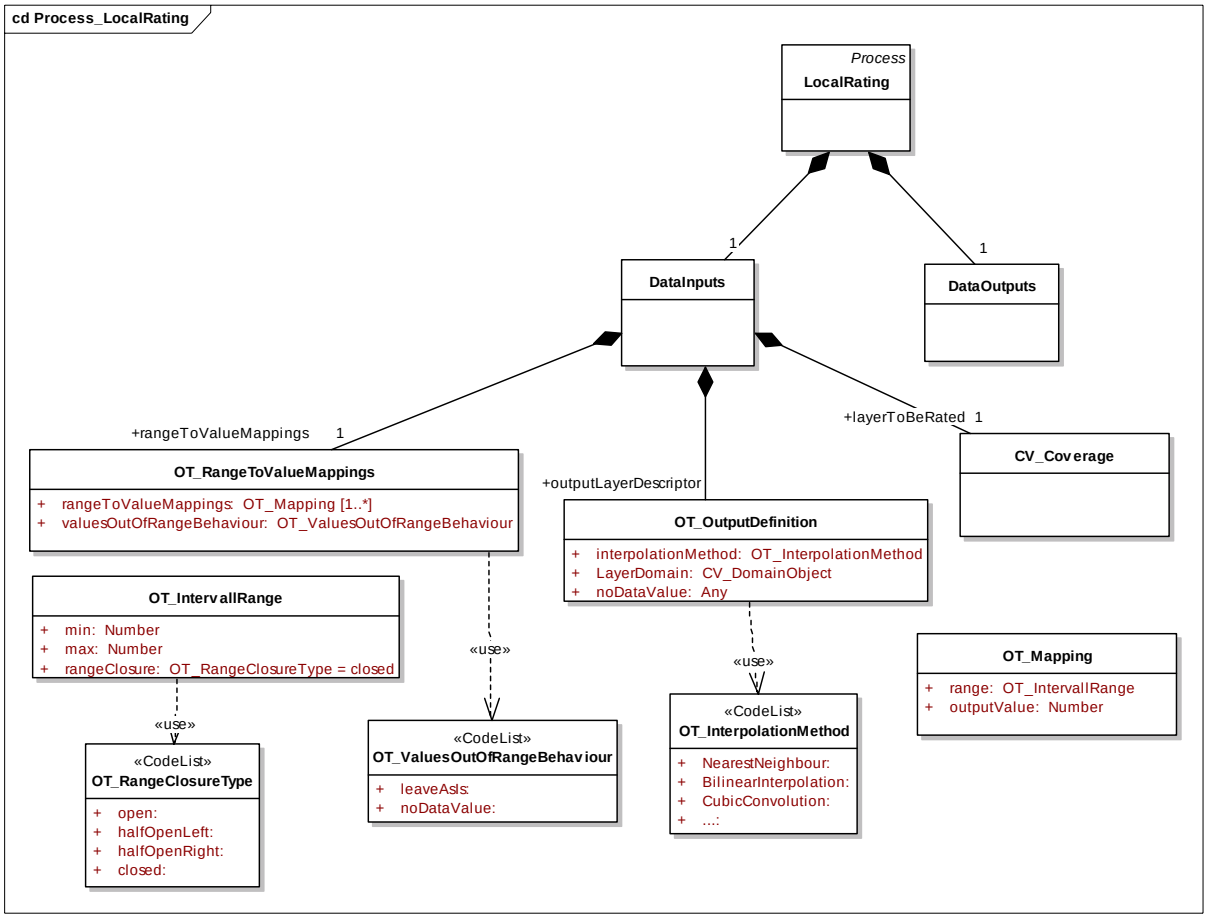


Figure 16 UML diagram illustrating the *LocalRating* Process

Figure 17 shows the "LocalRatio" process. It provides the ratio of each location's value of an input Coverage and a numeric value.

This process requires the following inputs: (1) an input Coverage that represents the dividend of the ratio (firstOperand), (2) a numeric value (of any type) that represents the divisor of the ratio (secondOperand) and (3) the definition of the output that determines the output Domain, the no-data value and the interpolation method that shall be used to create the output domain (if it differs from the domain of the input layer) (outputLayerDescriptor).

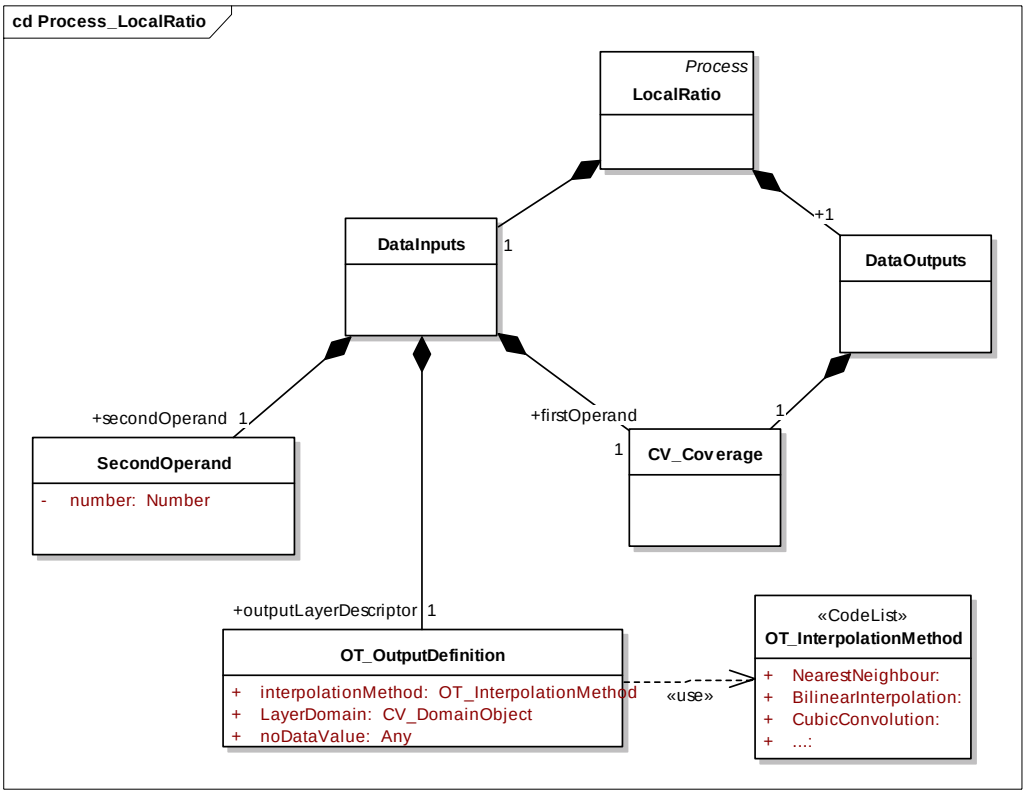


Figure 17 UML diagram illustrating the *LocalRatio* Process

In Figure 18 the "ZonalRating" process is described. It performs a reclassification and creates an output Coverage. The reclassification is based on the locations of an input Coverage that spatially correspond with the zones of a zonal Coverage.

The inputs of this process are as follows: (1) the input Coverage with values to be reclassified (valueLayer). (2) The zonal Coverage that provides the zones (zonalLayer); a zone shall be marked with a single numeric value. It is the equivalent of a polygon in the feature model. (3) The mapping rules for the reclassification (zonalRatingMapping). Like in the LocalRating process, there is an attribute that defines how to deal with those locations that are not defined in the mapping (valuesOutOfRangeBehaviour). The mapping itself allows to assign a new value to a combination of zone(s) of the zonal layer with value range(s) in the value layer. The new value might be either defined explicitly or refer to the corresponding value in either the zonalLayer or the valueLayer. (4) The definition of the output that determines the output Domain, the no-data value and the interpolation method that shall be used to create the output domain (if it differs from the domain of the input layer) (outputLayerDescriptor).

The output of the "Zonal Rating" process is a Coverage that contains the result of the processing.

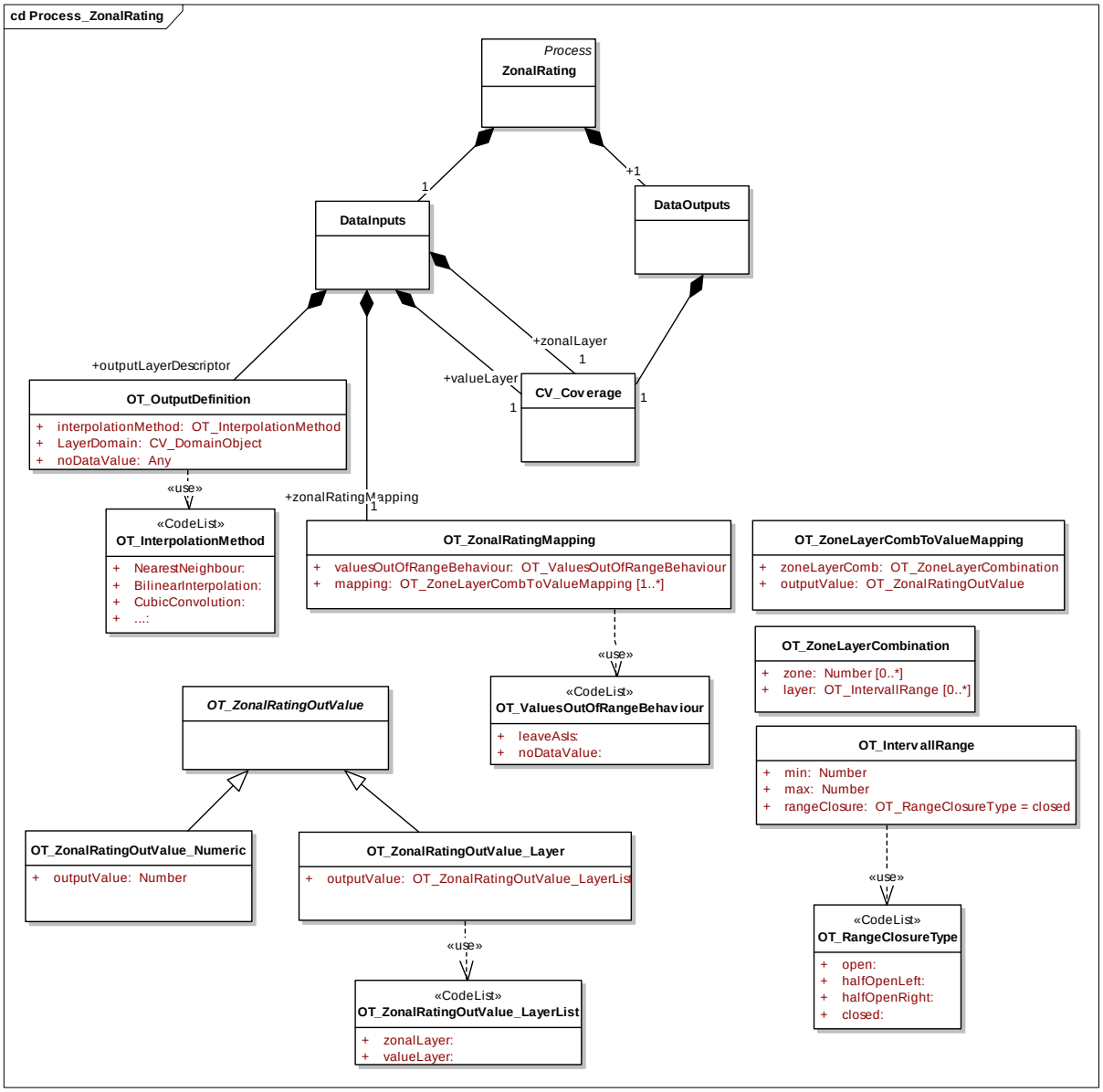


Figure 18 UML diagram illustrating the *ZonalRating* Process

Figure 19 shows the "ZonalSum" process. This process can be classified as an aggregation function. It computes the sum of all values of all locations of an input Coverage per zone of the zonal Coverage. The output sum per zone is linked to each location of he input Layer that lies in that particular zone.

The inputs of this operation are as follows: (1) The input layer whose location values shall be summed up (valueLayer). (2) The zonal layer (ZonalLayer). (3) The definition of the output that determines the output Domain, the no data value and the interpolation method that shall be used to create the output domain (if it differs from the domain of the input layer) (outputLayerDescriptor).

The output of the process is a Coverage that contains the result of the processing.

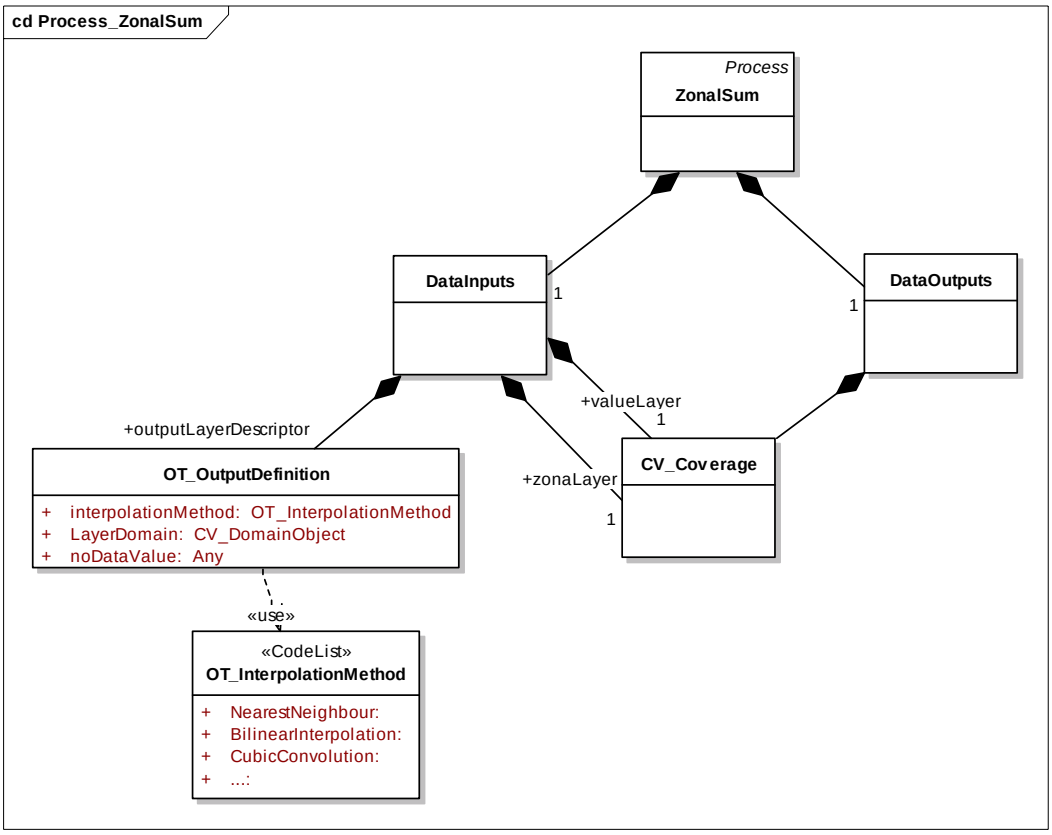


Figure 19 UML diagram illustrating the *ZonalSum* Process

3.4. OT Risk-specific services

Risk specific services are created for a certain purpose (e.g. forest fire risk assessment) and thematic community and specialised for a risk domain. Unlike the OT Risk-neutral services, they are not wrapped by the PS interface.

3.4.1. Forest Fire Risk Assessment interface

The forest fire risk assessment interface offers the following risk assessment functionalities: the frequency, the density and the levels of risk classification of forest fires based on administrative units, bounding box and time constraints.

The forest fire risk assessment interface offers 3 operations:

- **ForestFireFrequency**: the operation takes the information about selected administrative areas (by means of bounding box, feature types, etc.) and time constraints (i.e. for which time period the frequency shall be calculated) and returns the corresponding forest fire frequencies.
- **ForestFireDensity**: the operation takes the information about selected administrative areas (by means of bounding box, feature types, etc.) and time constraints and returns the normalized values of forest fires.
- **ForestFireRiskClasses**: the operation takes the information about selected administrative areas (by means of bounding box, feature types, etc.) and time constraints and returns the forest fire risk classification of the selected area.

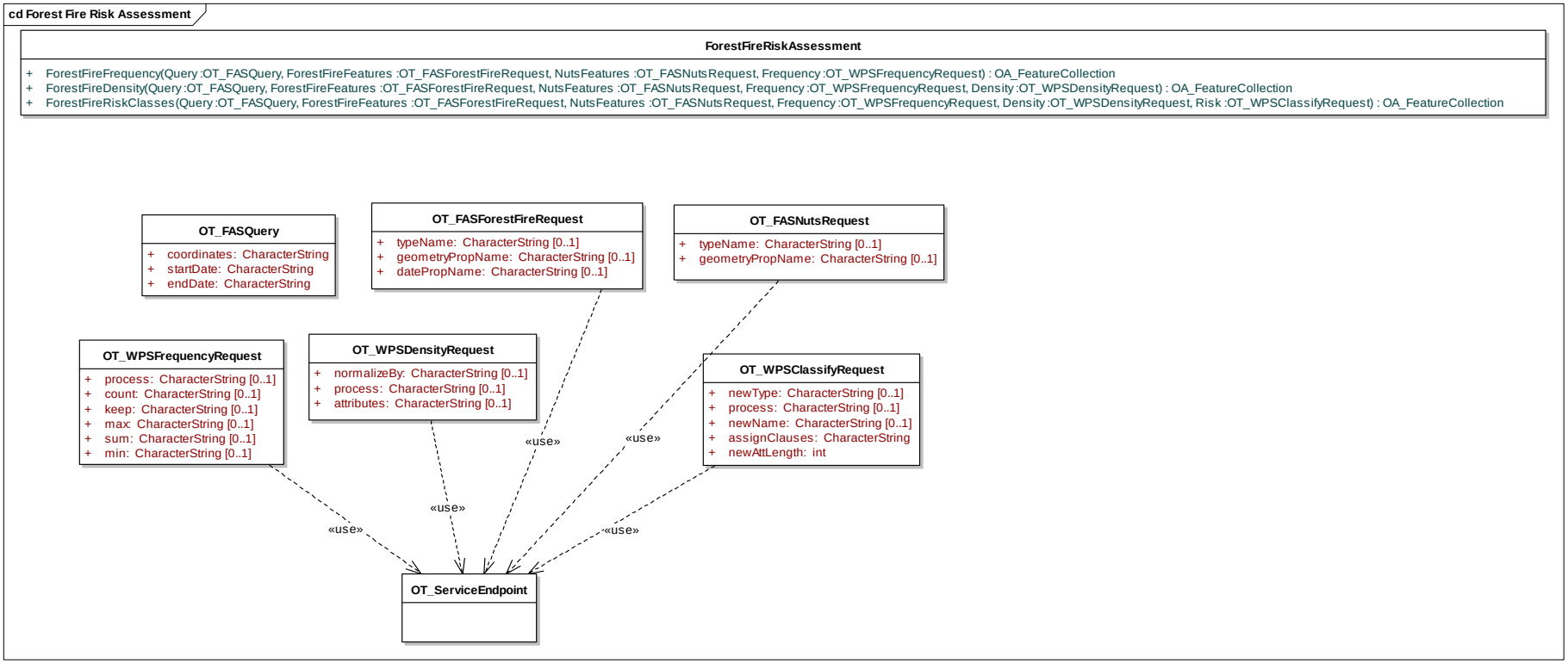


Figure 20: Conceptualisation of the operations offered by the Forest Fire Risk Assessment interface

3.4.2. Flood Simulation interface

The Flood Simulation interface provides a mean to simulate a flood for a given river and return the extent of this flood within the river's catchment area. The interface enumerates the rivers it supports.

The Flood Simulation interface offers one operation:

- **simulateFlood**: the operation uses the river name (river) and the flood height (floodHeight), and the definition of the output (outputLayerDescriptor, cf. Map Algebra interface). It returns a Coverage that represents the flood extent.

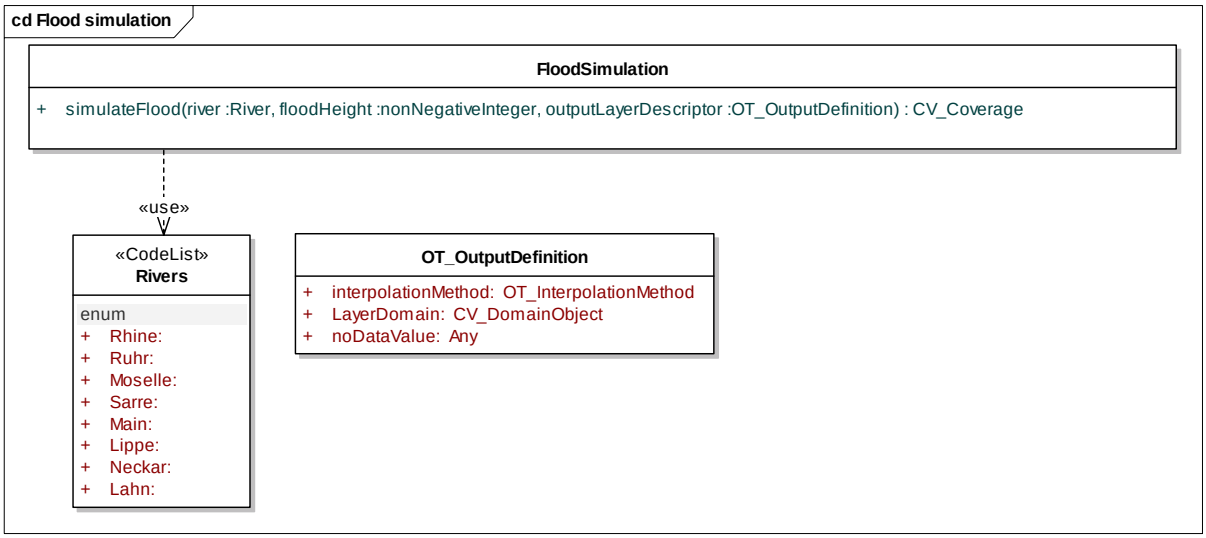


Figure 21: Conceptualisation of the operations offered by the Flood Simulation interface

3.4.3. Damage Assessment interface

The Damage Assessment interface offers means to assess the damage that is caused by any event that is described through its extent. The event extent has to lie within a certain bounding box for which the damage can be assessed. The available damage types and aggregation levels are specified in the interface. The output is a feature collection of features representing the aggregation level. These features have new attribute(s) representing the chosen damage.

The Damage Assessment interface offers one operation:

- **assessDamage**: the operation uses the damage type (damageType), the flood event (event), the events extent (eventExtent_BBOX) and a level for aggregation (aggregationLevel). It returns a feature collection.

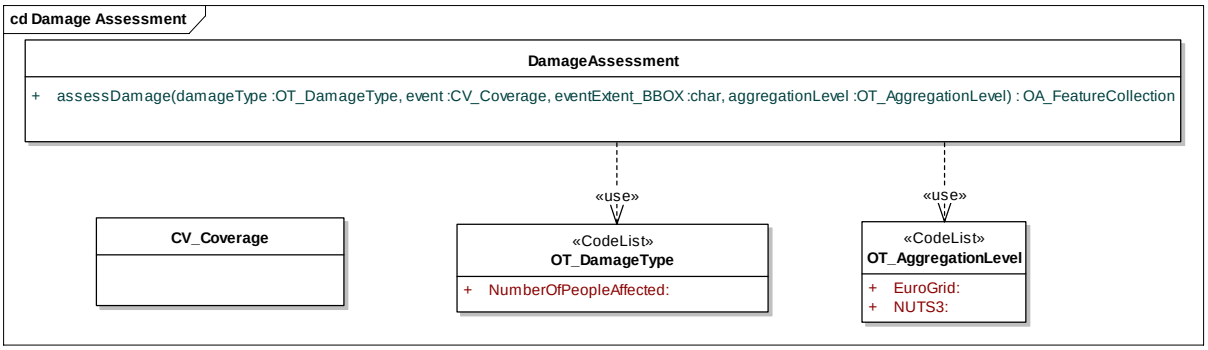


Figure 22: Conceptualisation of the operations offered by the Damage Assessment interface

3.5. Human Interaction Components

The ORCHESTRA Services do not provide an interface to a human user. This is covered by so-called Human Interaction Components (HCI), i.e. software components that provide the (usually graphical) user interface (GUI) of an OA Service or OT Service (e.g. Viewers and Editors).

3.5.1. Forest Fire Application and Map Viewer

Name	Forest Fire Application and Map Viewer	
Standard Specifications	*None*	
Description	This component provides the graphical user interface to the execution of the Forest Fire application domain and visualises the resulting map. Thus, it has to communicate with the component services. Figure 23 shows the dialogue for capturing the user input and initiating the workflow. Figure 24 shows the visualisation of the map resulting from the workflow execution.	
Operations	All operations are GUI operations. Inputs are provided troughs GUI controls (menus, dialogs, etc.). Outputs are marked as “NA” (Not Applicable) if the result is always visualization.	
Show Map	Description	Visualizes a map
	Input	Map provided by MAS
	Output	NA
Zoom in	Description	Visualizes the map as zoomed-in on the point or the box the user has clicked on
	Input	Point(s) (from Click-event)
	Output	NA
Zoom out	Description	Visualizes the Map as zoomed-out from the point or the box the user has clicked on
	Input	Point(s) (from Click-event)
	Output	NA
Pan	Description	Lets the user drag the map by pointing the mouse on it
	Input	Point (from Click-event)
	Output	NA
Zoom to full extent	Description	Visualizes the map as zoomed in order to show the complete map
	Input	-
	Output	NA
Back/Forth to previous/next extent	Description	Changes visualisation to previous or next extent in the map viewer history
	Input	-
	Output	NA
Visualize coordinates	Description	Visualizes the coordinates of the point where the mouse is on. This is a continuous functionality that does not have to be explicitly chosen by the user, i.e. the mouse pointer position is always displayed (in map coordinates).
	Input	Mouse position
	Output	NA

Show feature info	Description	visualizes the attributes concerning the features individuated by the click of the user
	Input	Point (from click event)
	Output	NA
Show selected features	Description	Highlights the features individuated by the click of the user or selected by a query
	Input	Point(s) or box (as from click events)
	Output	NA
Show/hide legend	Description	Visualizes/hides legend symbols
	Input	-
	Output	NA
Change symbology	Description	Changes the symbology for the map visualisation
	Input	User selections and text input (see Figure 24)
	Output	Description of selected symbology
Select statistics	Description	Selects the type of statistics to be returned by the processing service
	Input	User's selection on a combo-box
	Output	Selected type of statistics
Choose spatial aggregation level	Description	Selects an aggregation level based on the selection by the user
	Input	User selection in radio button group
	Output	Selected aggregation level
Choose temporal extent	Description	Selects a temporal extent based on the input by the user
	Input	User input in text fields
	Output	Selected temporal extent
Select spatial extent	Description	Selects a spatial extent based on graphical selection (bounding box or sequential selection of features) by the user
	Input	Points (from click event)
	Output	Selected spatial extent
Execute workflow	Description	Executes the workflow and displays the resulting map
	Input	User-selected parameters (see above)
	Output	Result of the service chain (map served by MAS)
Example usage	See D4.2.1, chapter 5.	
Comments	The map format can be either raster (e.g. png, jpeg) or vector (e.g. SVG, pdf) (see MAS specification). For each interaction with the User, the Forest Fire Application client needs to detect the coordinates of the point within the canvas where the user has clicked on. These canvas coordinates need to be translated into map coordinates.	

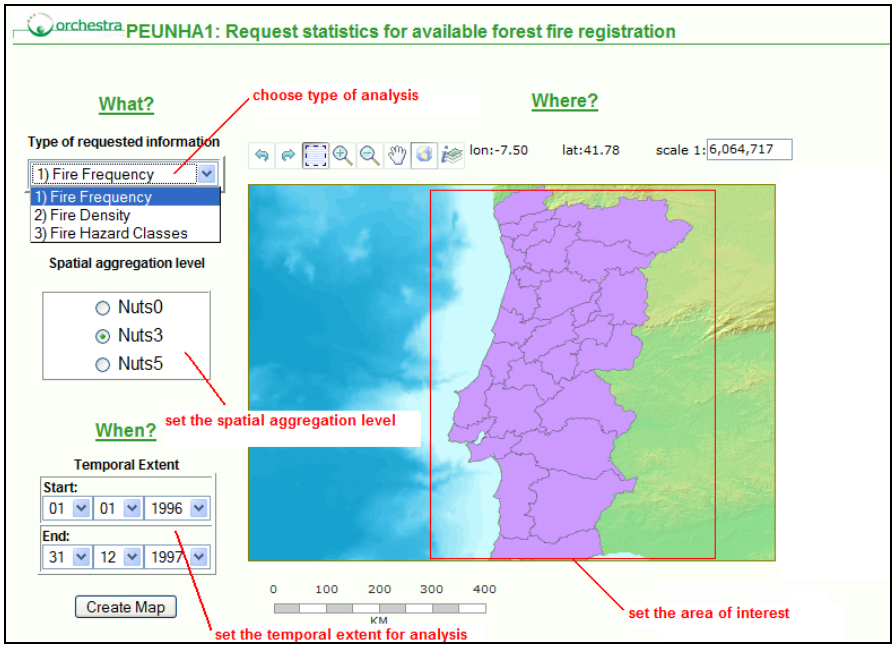


Figure 23: Screenshot of the client interface – Request forest fire analysis.

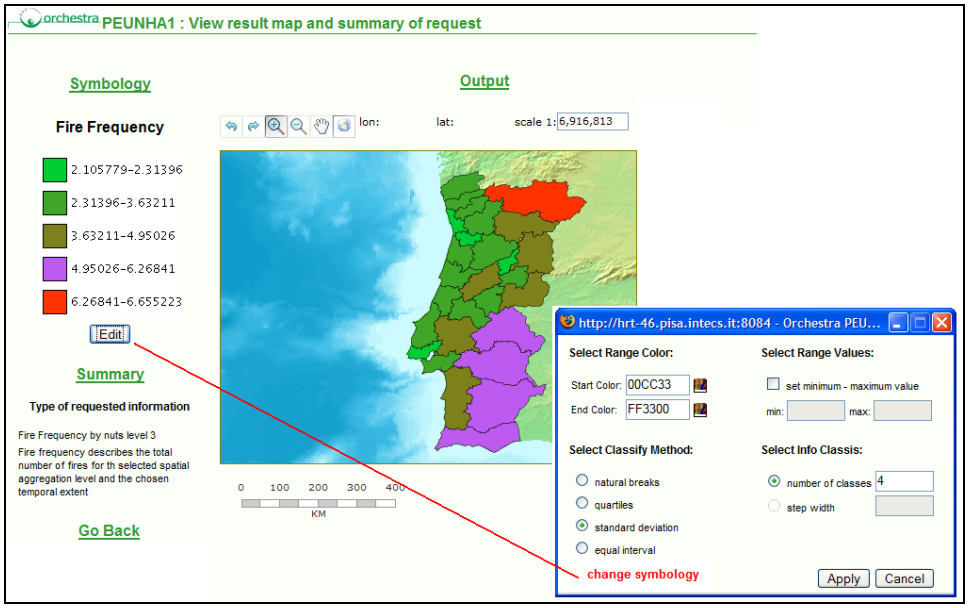


Figure 24: Screenshot of the client interface – View result map (metric attribute).

3.5.2. Flood Simulation & Damage Assessment Application

Name	Flood Simulation & Damage Assessment Application Client and Map Viewer
Standard Specifications	*None*
Description	*Note: This application is currently still under development.* This component provides the graphical user interface to the execution of the service chain defined in the Flood Simulation and Damage Assessment application domain and visualises the resulting map. Figure 25 shows the dialogue for capturing the user input and initiating the workflow. Figure 26 shows the visualisation of the map

	resulting from the workflow execution.	
Operations	All operations are GUI operations. Inputs are provided troughs GUI controls (menus, dialogs, etc.). Outputs are marked as "NA" (Not Applicable) if the result is always visualization. All operations related to the main map visualisation[6], as well as the "Change symbology" operation, are available in this application as in the previously described Forest Fire Application (cf. section 3.5.1). Other operations are described here below.	
Select the river basin	Description	On a list of available datasets, the user can choose the river for assessment.
	Input	User's selection on a combo-box
	Output	Selected river basin / geographic extent for analysis
Define flood level	Description	On a list of values, the user defines the flood level.
	Input	User's selection on a combo-box
	Output	A flood level
Choose spatial aggregation level	Description	Selects an aggregation level based on the selection by the user
	Input	User selection in radio button group
	Output	Selected aggregation level
Define damage indicator	Description	The user selects the indicator to be used in damage assessment from a list, such as number of people affected.
	Input	User's selection on a combo-box
	Output	Selected indicator
Execute workflow	Description	Executes the workflow and displays the resulting map
	Input	User-selected parameters (see above)
	Output	Result of the service chain (map served by MAS)
Example usage	See D4.2.1, chapter 5.	
Comments	The map format in raster format (e.g. png, jpeg).	

[6] I.e: "Show map", "Zoom in", "Zoom out", "Pan", "Zoom to full extent" "Back/Forth to previous/next visualisation", Visualise coordinates" and "Show feature info"

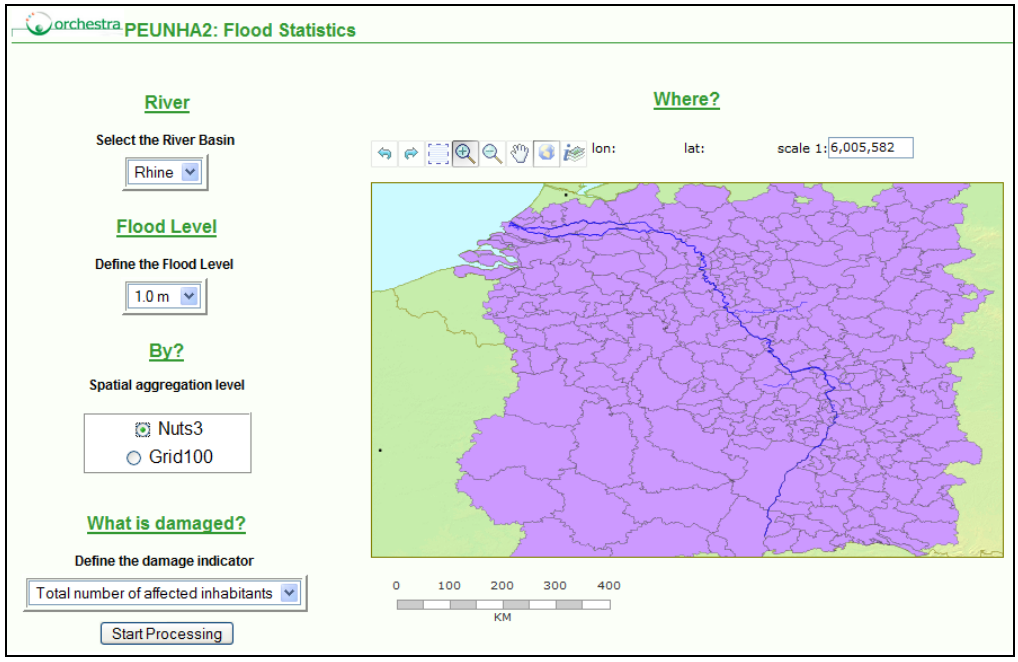


Figure 25: Screenshot of the client interface – Request Flood hazard assessment.

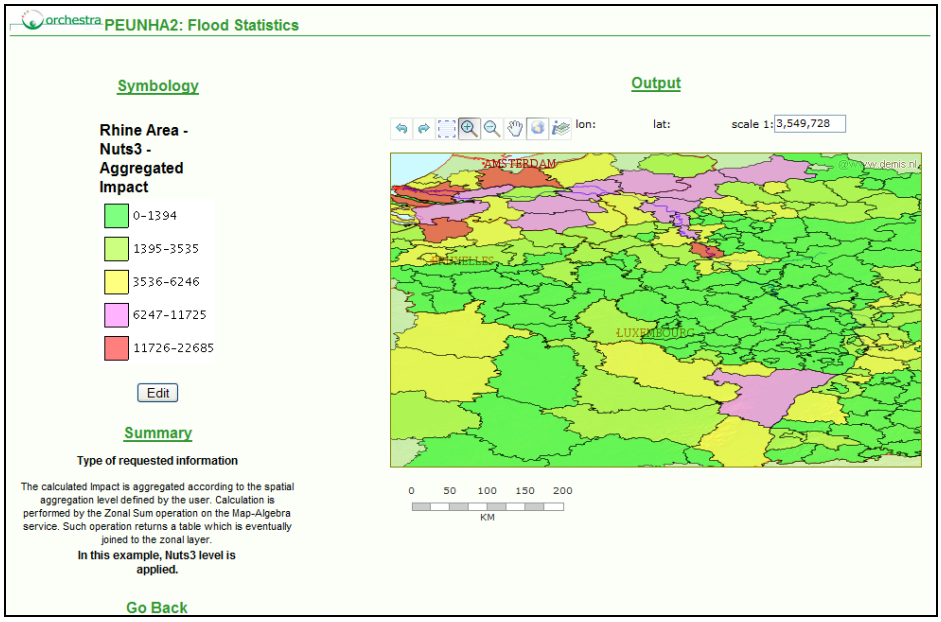


Figure 26: Screenshot of the client interface – View result map.

4. Implementation: The ORCHESTRA Service Network

This section describes the implementation of the two pilot applications for forest fire risk assessment and flood-related damage assessment. Section 4.1 introduces the platform used for the implementation. Section 4.2 describes the implementation architecture. The client implementations are discussed in sections 4.2.4 and 4.4. The service chains implementing the actual applications and the services they use are described in section 4.3.

4.1. Platform specification

The specification is a combination of the ORCHESTRA Web Services Platform, which is an adaptation of the W3C Web Services Architecture. It specifies the use of the Hypertext Transfer Protocol version 1.1(HTTP 1.1)[7] and SOAP version 1.2 (SOAP 1.2)[8] as protocols for the exchange of structured information, which is encoded using the eXtendible Markup Language version 1.0 (XML 1.0)[9] and XML Schema Part 1 and 2 (XML Schema)[10]. The XML based Web Services Description Language Version 1.0 (WSDL 1.0)[11] is used to describe Web Services. For the encoding of spatial data, the platform specification foresees the use of the Geography Markup Language (GML). GML is an XML grammar written in XML Schema for the modeling, transport, and storage of geographic information. The key concepts used by GML to model the world are drawn from the OpenGIS® Abstract Specification and the ISO 19100 series. In the ORCHESTRA Web Services Platform they foresee the use of GML version 2.1.2 (GML 2.1.2)[12] and version 3.1.1 (GML 3.1.1)[13]. As an alternative encoding for Grid Coverages, the platform specification foresees the use of GeoTIFF, version 1.8.2 (GeoTIFF)[14]. GeoTIFF is a TIFF based interchange format compliant with TIFF 6.0. The GeoTIFF specification de-fines a set of TIFF fields that allow to geo-reference a TIFF. For the full specification, see [11].

4.2. Pilot Architecture overview

In this section we illustrate the ORCHESTRA Service Network (i.e. the actual instances of the previously specified components) for the two application areas: i.e. the distributed services and client components and their relationships (i.e. the interactions occurring between them). We distinguish four sections for the three application areas and the corresponding use case(s): Application Development, Forest Fire Risk Assessment, Flood Damage Assessment and Schema Transformation.

4.2.1. Application Development

The idea of workflow and services able to execute and control workflows has been applied in the development of the pilots. In particular, the programming pattern that has been used consists of creating workflows equipped of operations allowing the client application (i.e. the one interacting with the human users) to directly invoke one single operation for each of the supported functionalities that, in our case, are the different risk analysis the user may want. The advantages of such pattern are the complete separation of what functionalities are necessary from how they are actually implemented. The client application and the service executing the workflow are coupled via a contract defining the operations that are supported and the way to invoke and handle them. The concrete implementation of the workflow on the other hand is totally hidden to the client that consequently does not to deal with all those aspects related with the concrete network of services. In terms of sustainability, one important benefit is that any change at the services network that may affect the workflow definition will not

[7] http://www.w3.org/Protocols/rfc2616/rfc2616.html

[8] http://www.w3.org/TR/2004/NOTE-soap12-af-20040608/

[9] http://www.w3.org/TR/2004/REC-xml-20040204

[10] http://www.w3.org/TR/xmlschema-1/

[11] http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[12] http://portal.opengeospatial.org/files/?artifact_id=11339

[13] ISO/CD 19136 Geographic information - Geography Markup Language (GML), Version 3.1.1, http://portal.opengeospatial.org/files/?artifact_id=4700

[14] http://www.remotesensing.org/geotiff/spec/geotiffhome.html

imply any change at the client applications: the client will continue to rely on a single service executing the workflow, which is the single service aware of the business logic for functionalities implementation.

The Service Chain Access Service (SCAS) is a service that creates an executable service instance based on an explicit description of a service chain. The service chain, expressed through a workflow language (in the specific case in Business Process Execution Language), can then be executed as a single service. SCAS supports, through the createServiceChain operation, a service provider in creating an executable instance of an aggregate service based on an explicit service chain description defined by the service provider. Optionally it is also possible to register the created service chain instance in a catalogue service so that it can be discovered as the other available ORCHESTRA services (however this option has not been implemented).

The following two figures show the procedure for implementing this programming pattern. Once the workflow has been defined, the service integrator uses the SCAS client for creating a new service instance, in the example the one responsible of offering all the risk analysis for the Forest Fire Risk Assessment pilot. Once the service executing the workflow is available in the network, it can be used by client applications that will deal only with the Risk Assessment service. The execution of the service chain thus corresponds to the call of an ORCHESTRA Service Instance operation in an ORCHESTRA Service Network.

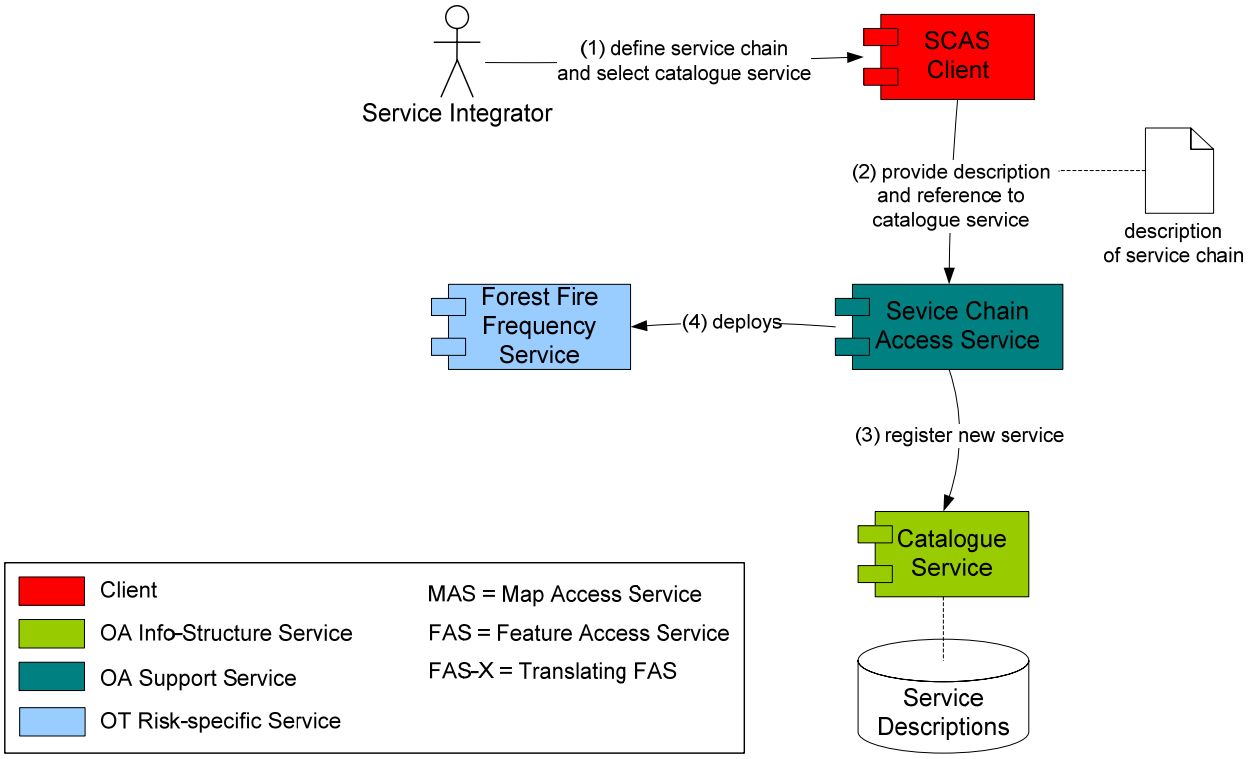


Figure 27: Exemplary deployment of a service chain description

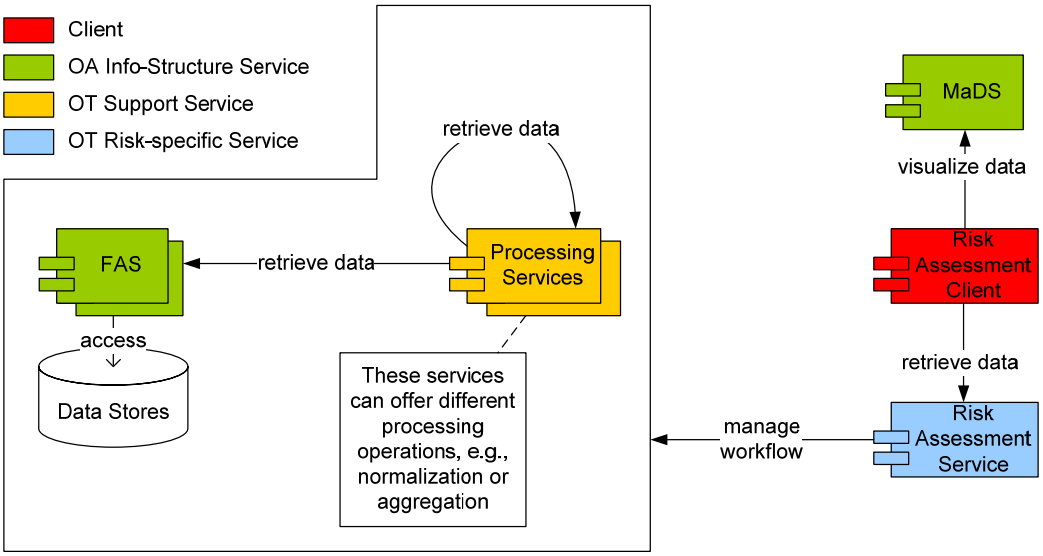


Figure 28: The proposed architecture for supporting application development based on service composition.

The following sections are devoted to describe the specific workflows supporting the identified thematic use cases, focusing on the functional and composition aspects through the description of the single services and corresponding functionalities that have been composed in a workflow.

Of course when dealing with applications distributed over the Internet additional aspects play a crucial role, such as the quality of service in terms of scalability and performance. The most relevant related issues pertain to service interoperability (e.g. interfaces) and data passing modalities (by value or by reference) used to exchange data between services with implications on control and data flow patterns to be used to program the workflow. For a detailed analysis of such aspects, with explicative examples based on the Forest Fire Risk Assessment pilot, see [12].

4.2.2. Forest fire risk assessment application

The conceptual workflow and the data flow of the forest fire risk assessment application are depicted in Figure 29, while the architecture that implements the conceptual workflow is described in Figure 30. The member state forest fire registration data are accessed through a FAS-X[15] that maps the local schemas to a common schema that is used for further processing. The Normalisation Service is only used for the ForestFireDensity and ForestFireRiskClasses operations, the Classification Service only for the ForestFireRiskClasses operation.

[15] The FAS-X implements the FAS interface (which is classified as OA Info-Structure) as well as the Schema Mapping Repository (which is classified OA Support). Hence, it is marked in both colors inFigure 29.

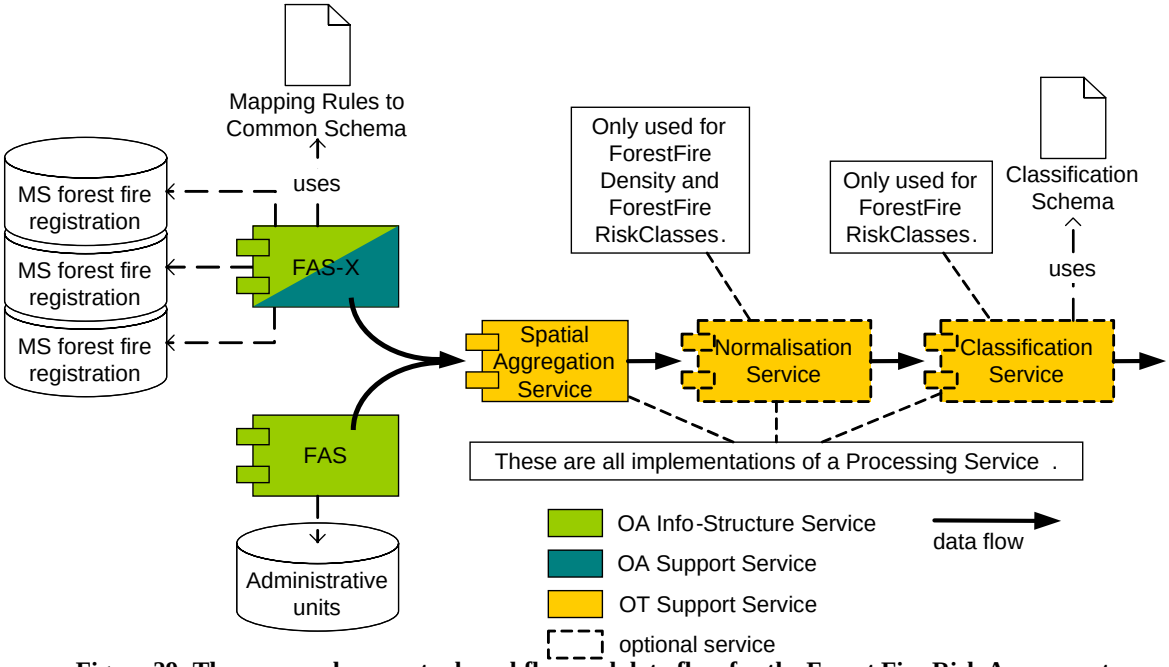


Figure 29: The proposed conceptual workflow and data flow for the Forest Fire Risk Assessment.

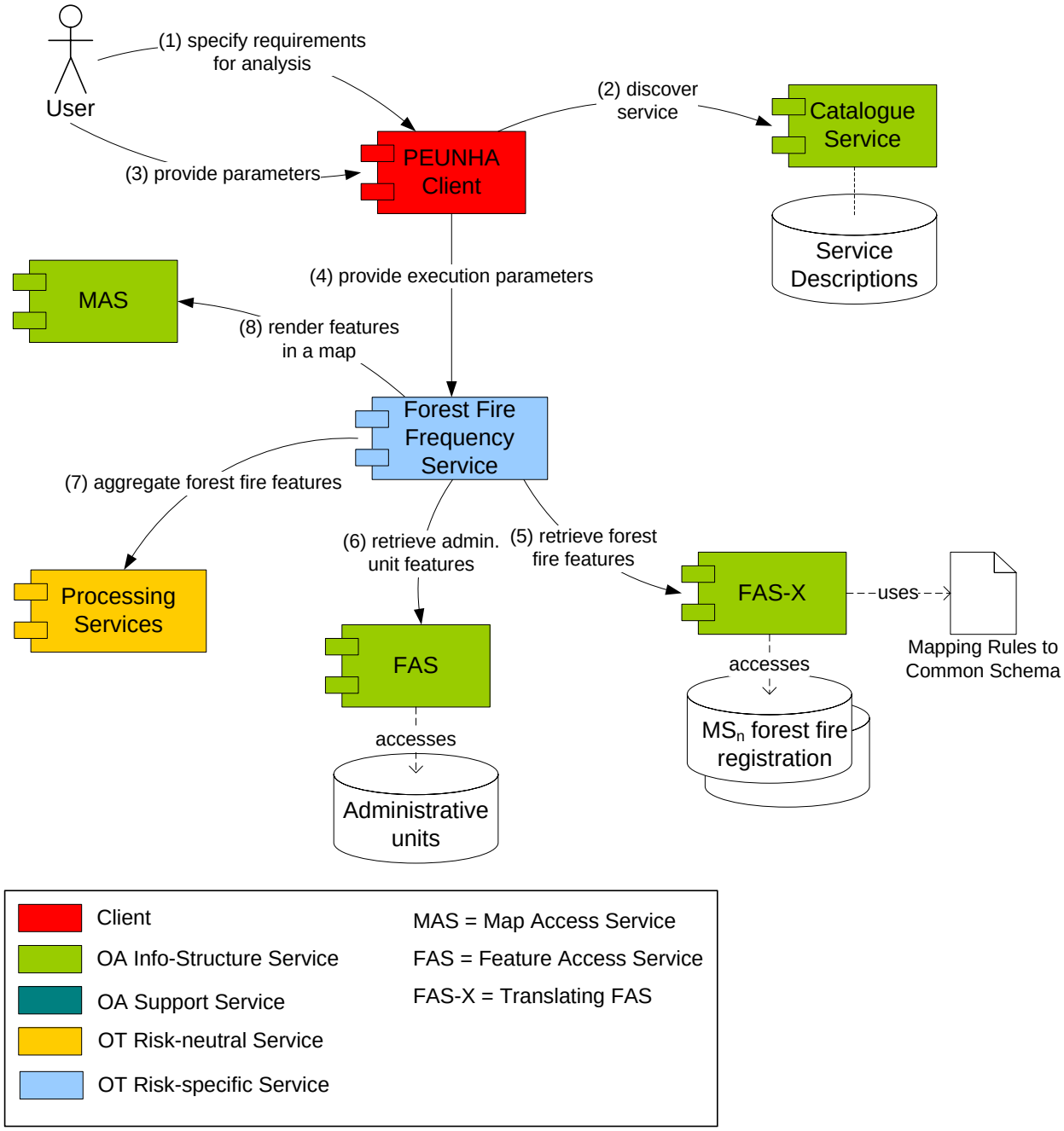


Figure 30: Architecture for the Forest Fire Risk Assessment Application that implements the conceptual workflow

4.2.3. Flood damage assessment application

The conceptual workflow and the data flow of the flood damage assessment application is depicted in Figure 31 (flood extent simulation) and Figure 32 (impact or damage assessment), while the architecture that implements the conceptual workflows is described in Figure 33 and Figure 34.

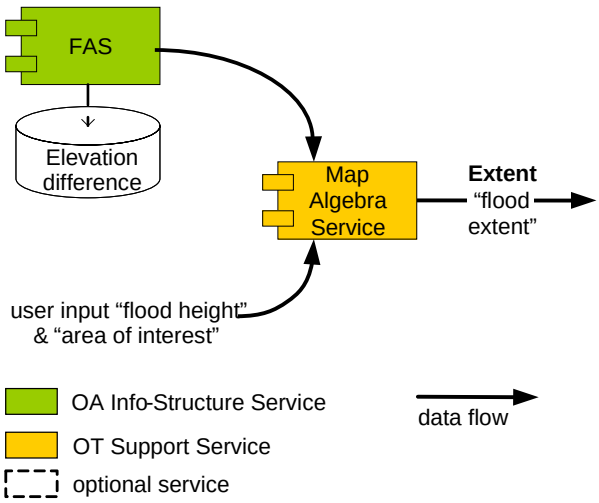


Figure 31: The proposed conceptual workflow and data flow for the Flood Damage Assessment – Part 1: Flood extent simulation.

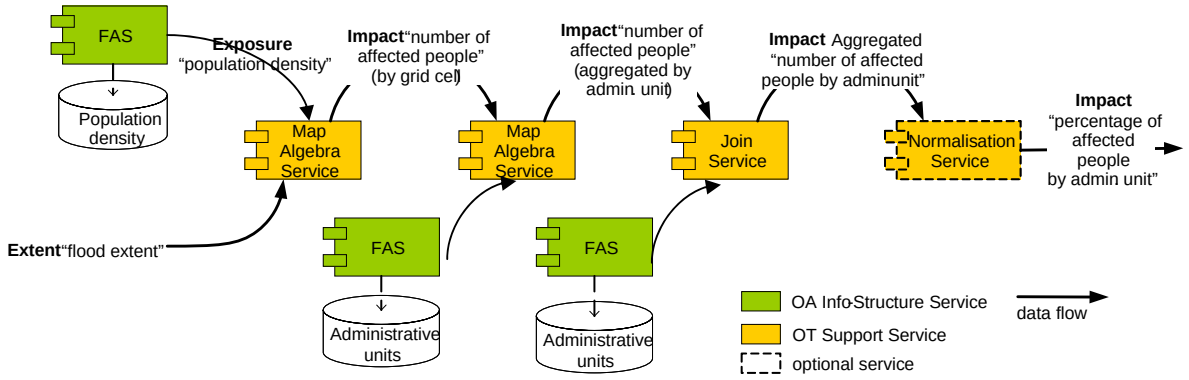


Figure 32: The proposed conceptual workflow and data flow for the Flood Damage Assessment – Part 2: Impact assessment.

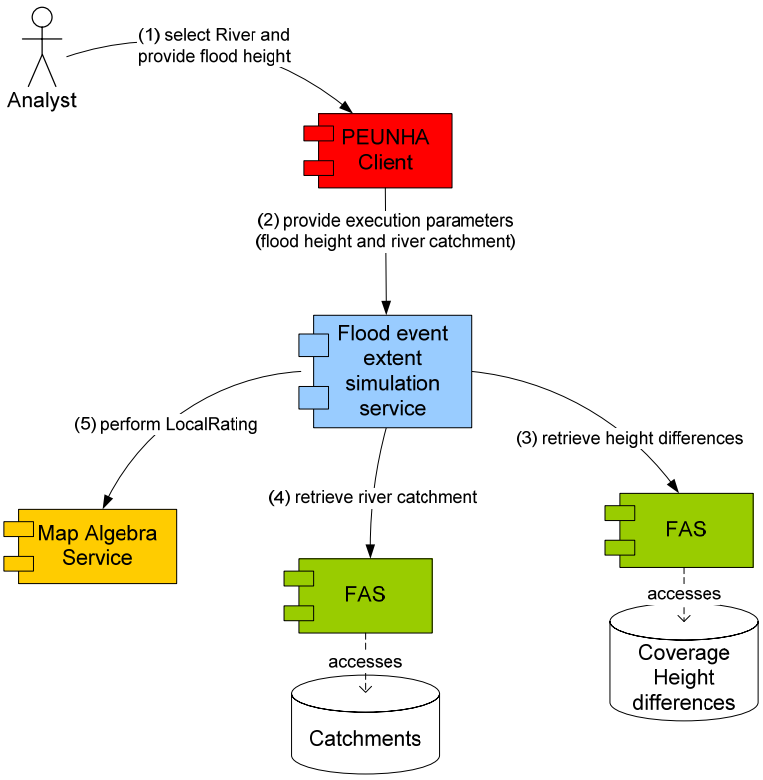


Figure 33: Architecture for the Flood Risk Assessment Application that implements the conceptual workflow for the flood extent simulation.

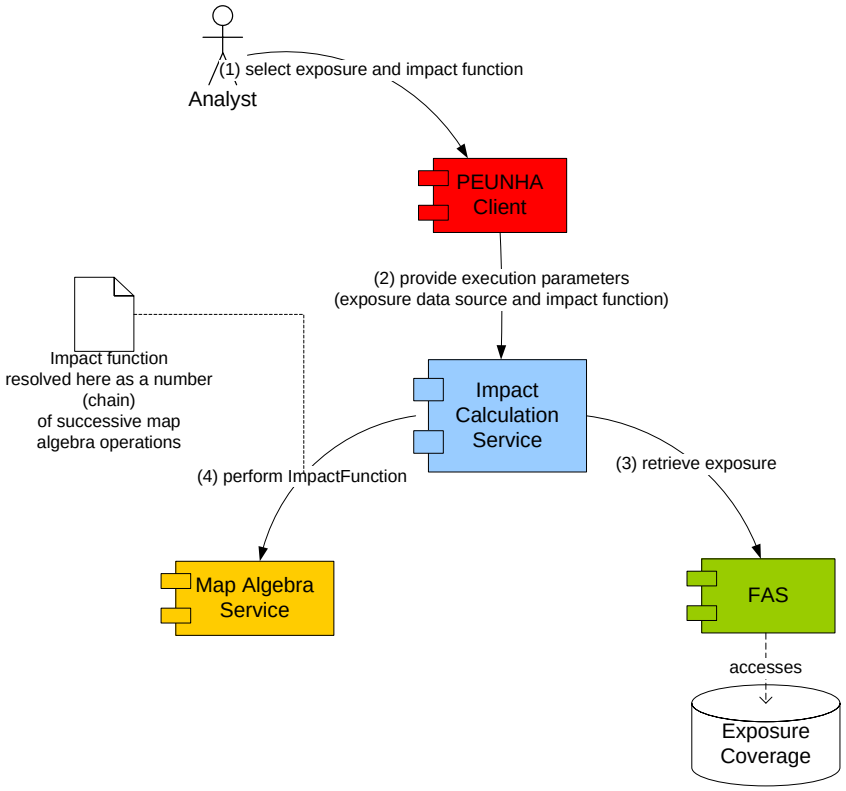


Figure 34: Architecture for the Flood Risk Assessment Application that implements the conceptual workflow for the damage assessment.

4.2.4. Schema Mapping Application

The schema mapping application shall allow the registration of mappings from a source into a target schema, so that forest fire records that are stored in their local schema can be accessed in a harmonised common schema. In the ORCHESTRA Service Network architecture, this functionality is provided through the FAS-X that allows registering mappings from a source to a target schema for registered Feature Collections (Figure 35) and accessing these Feature Collections in the common schema, which are then transformed on the fly (Figure 36).

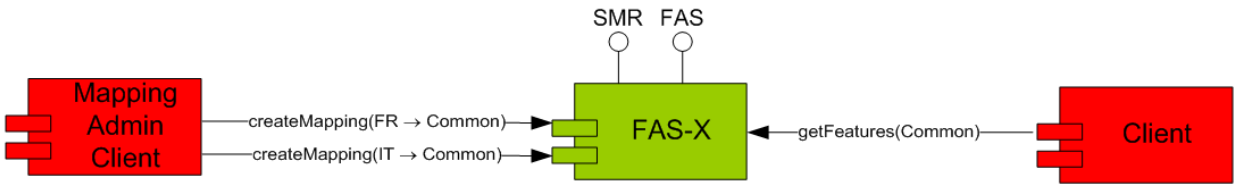


Figure 35: Registering a new mapping with the FAS-X

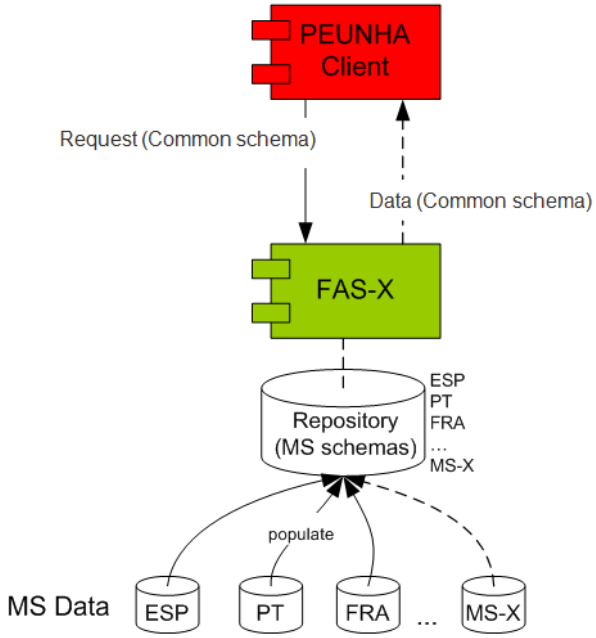


Figure 36: Accessing harmonised data - request MS forest fire data stored in local schema with common schema

4.3. Service Instances

The following sections are devoted to describe the most relevant implementation specific details for each used service type and corresponding functionalities.

4.3.1. SCAS

The SCAS implementation supports the WS-BPEL language for defining concrete service chains. To this end the ActiveBPEL engine (www.activebpel.org) and the Web services it provides for dealing with administration functionalities (e.g. for deploying a new BPEL process) are used. ActiveBPEL is distributed as an Open Source License (www.active-endpoints.com/open-source-license.htm).

4.3.2. Feature Access Services (FAS)

The FAS provides access to data on administrative units (NUTS), population density, the Rhine catchment and the height difference to the closest river. The FAS is implemented as a wrapper to existing OGC WFS/WCS instances. In the pilot applications, a GeoServer WFS (version 1.5.3) is used as an underlying service for vector data and a GeoServer WCS (version 1.5.6) as an underlying service for coverage data. Both are distinguished through the used query language: When *get_wfs* or *post_wfs* is used this is interpreted as a request for vector data, while the usage of *get_wcs* or *post_wcs* identifies a request for coverage data:

When the FAS receives a *getFeatures* request with the query language attribute set to *get_wfs/post_wfs*, it generates a WFS *getFeature* request (using HTTP POST) with a corresponding OGC filter expression and sends this to the underlying GeoServer WFS. The WFS responds with a GML Feature Collection, which is passed on in form of a String in the body of the SOAP message as the FAS response.

When the FAS receives a *getFeatures* request with the query language attribute set to *get_wcs/post_wcs*, it generates a WCS *getCoverage* request (using HTTP GET or POST) and sends this to the underlying GeoServer WCS. The WCS responds with a Binary file. This file is transformed into a base64-encoded String and passed on in form of a String in the body of the SOAP message as a FAS response.

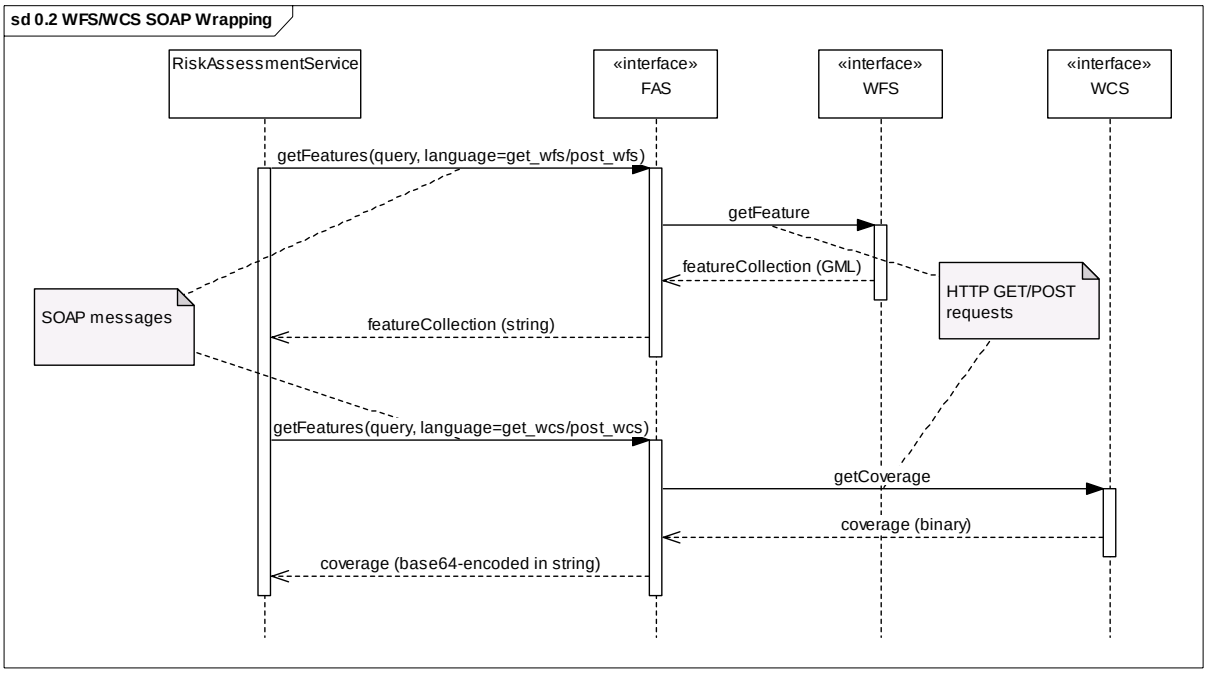


Figure 37: Sequence diagram illustrating how WFS and WCS instances are wrapped by a FAS.

4.3.3. Translating FAS (FAS-X)

The FAS-X provides access to member state forest fire registration data including the information on the location of the assumed ignition point (as geographic or projected coordinates or geographic identifier), the date and time of ignition, the burnt area (including the fraction for “forest and other wooded land area” and “non forested area”) and the fire cause. It maps data from their local schema (the source schema) to the target schema (as proposed in the implementation rules to the Forest Focus Regulation (EC No 2152/2003) in the forest fire application). The mappings from the local to the common schema are pre-defined in the FAS-X. Additionally, the service provides the possibility to store all registered mappings in the Schema Mapping Repository for later consultation and modification. When a new mapping is created, its data source is mapped to the common target schema and the mapping descriptor is stored automatically in the repository. Then the data source is published in the common schema and the mapping descriptor becomes available for retrieving and modifying.

The FAS-X is implemented as a wrapper to a Snowflake GO Publisher WFS. The Snowflake GO Publisher WFS[16] is a Translating WFS that maps data from a source schema into a target schema on the fly using a pre-defined mapping. The Snowflake GO Publisher WFS supports a proprietary mapping language, which is also used in the other GO Publisher products. The underlying Translating WFS can be created using the Go Publisher Desktop (version 1_3_0j). This service contains a mapping document (in XML format) that represents all of the available mappings of the FAS-X. Whenever the FAS-X receives a createMapping or a deleteMapping request it accesses to the mapping document of the Translating WFS, modifies it remotely through the GoPublisherHelper servlet application (developed by JRC). The Tomcat Server (version 5.5) takes a few seconds to refresh the Translating WFS to publish the new data source into the common target schema so that it can be queried by the FAS-X. There are some issues that have been experienced in using the Translating WFS of Snowflake as an underlying service. These issues will be discussed in detail in section 6.1.9. It is necessary to store the processing data of the service in a data base. The PostgreSQL database (version 8.1) is used for that purpose.

4.3.4. Schema Mapping Service

The Schema Mapping Service Instance is not part of the pilots' information flow but was implemented and tested as an alternative approach to the FAS-X.

The Schema Mapping Service provides the functionality to map a feature collection encoded in a source schema into another feature collection encoded in a target schema. The transformation is performed according to a mapping rule that either has been registered in the service local repository (through the operations specified in the Schema Mapping Repository Interface), or that is specified on-the-fly.

The mapping script can be expressed in different languages. The current implementation supports XSLT 1.0 and XSLT 2.0 through different implementation engines. The XML transformation engine implementations used are:

- XSLTC for XSLT 1.0 style sheets. XSLTC is considered to be the best implementation choice of the two transformation engines provided by JAXP, in term of memory requirements and performance for large and complex data transformation.
- Saxon-B and Saxon–SA for XSLT 2.0 style sheets. Saxon-SA is the schema-aware commercial version of Saxon-B and contains Saxon-B. The schema-aware functionality of the engine is activated if a valid license is present.

If no valid license is present the following functionalities are not available:

- the validation of the source feature collection against the source schema
- the validation of the output feature collection against the target schema
- the streaming optimization

XSLT transformation engines usually consume large amounts of central memory. One closed-source, commercial implementation (Saxon SA), offers an optimization applicable to the case when a feature type must be transformed into another feature type whose attributes can be determined exclusively from the values of the source feature type. The attempt of computing document scope values (e.g. totals or group values) breaks the optimization.

The optimization is triggered by the following syntax:

```
<xsl:function name="f:extractFeatureMembers">
   <xsl:copy-of select="doc($featuresfilename)//(gml:featureMember|gml:boundedBy)" saxon:read-once="yes"
xmlns:saxon="http://saxon.sf.net/"/>
</xsl:function>
```

Feature members are extracted from the source document and are not stored in memory.

[16] http://www.snowflakesoftware.co.uk/products/gopublisher/wfs/

4.3.5. Repository Service

The purpose of the Repository Service is to make results of a FAS request available through a URL. Two types of this service are used in the pilot, one for storing GML data, and one for storing GeoTIFF. Both types receive a string (containing the GML or the base64-encoded GeoTIFF) as input, store the data locally and make it accessible through a URL, which is returned as a result.

The solution of using two types of Repository Service (rather than one that includes the format as an additional input parameter) was chosen for simplicity. The service only becomes necessary because the Processing Services used in the service chain instances require their input data to be passed by (URL) reference rather than by value.

4.3.6. Processing Service (PS)

A Processing Service provides a range of different processing algorithms such as normalization or map algebra, through a single interface (cf. RM-OA v. and D4.2.2). The interface provides three generic operations *getCapabilities*, *describeProcess* and *execute*, while each algorithm is called through a parameters given as input to the execute operation. This strategy is inline with the OGC specification for the Web Processing Service (OGC WPS). The Processing Services are implemented using a wrapper to underlying OGC WPS service instances. Two such wrappers have been developed, one based on AXIS2 and one based on Intecs SSE Toolbox. The SSE Toolbox (http://toolbox.pisa.intecs.it) is a configurable Web Application that provides a SOAP interface to different type of services. The core of the Toolbox is an XML engine that interprets XML scripts (valid according the Toolbox XML script schema): such scripts represent the glue between the Toolbox and the service requiring the SOAP interface. Within the PEUNHA pilot, the Toolbox has been used to provide a SOAP interface to two processing services: the Classification Service and the Map Algebra. Any time it receives a *getCapabilities*, *describeProcess* or an *execute* request, the Toolbox:

- extracts the content from the payload of the SOAP request
- using such content, performs a POST request to the addressed service
- envelopes the HTTP response in a SOAP response.

Figure 38 shows how a risk assessment service requests a PS instance using SOAP messaging and how the request is passed on to the underlying WPS instances using HTTP POST.

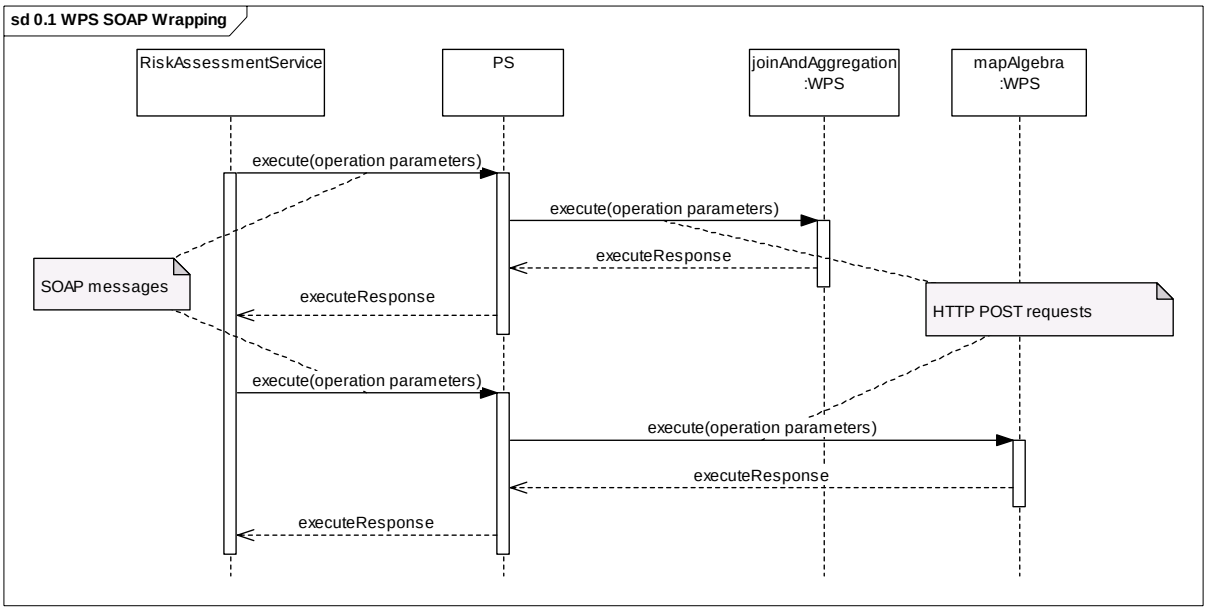


Figure 38: Sequence diagram illustrating how different WPS instances are wrapped by a PS.

The PS instances and the algorithms they provide are listed in the following sections. Each instance provides a list of related algorithms, in the following referred to as processes or processing operations, and has been named

accordingly. In the future, the grouping of related processes into processing service instances could be based on a taxonomy for geoprocessing operations.

4.3.6.1. PS Join-Aggregate-Normalise Service instance

This Processing Service Instance has been implemented based on the 52N WPS Framework[17]. To implement the processing algorithms, the GeoTools API (version 2.4) is used. The instance provides processes for joining, aggregating and normalizing GML Feature Collections. The ProcessDescription documents describing the processing operations of the service are given in Appendices **A.1.1** to **A.1.3**.

4.3.6.2. PS Classification Service

The Classification Service provides a list of algorithms that handle geographic features; it provides two main types of functionalities: it can be used in order to get a classified map or to add and populate a new attribute to a feature-type. In both cases, the user, after specifying the numeric attribute(s) to be used in the classification process, can choose to perform a standard classification (applying common classification criteria, such as quantiles or natural-breaks) or can specify a customised classification method, by means of logical clauses.

The Classification Service has been implemented by means of GRASS (version 6.2) libraries. The ProcessDescription documents describing the processing operations of the service are given in Appendix **A.2**.

4.3.6.3. PS Map-Algebra Service

The Map-Algebra service implements a subset of the operations specified as Map algebra. Each operation generates new coverage data by processing available GeoTIFF or ASCII Grid data. Furthermore, each processing operation embeds a set of functions which allow users specifying the geographic domain, the coordinate system and the format of resulting data.

Most processing operations of the service have been implemented by means of the GRASS Mapcalc tool [13]. The ProcessDescription documents describing the processing operations of the service are given in Appendix **A.3**.

4.3.7. Map and Diagram Service (MAS)

The implementation of the MAS is described in [14]. The MAS is available as an open source implementation at http://karlinapp.ethz.ch/qgis_wms/index.html.

4.3.8. The PEUNHA Service Chains

The following sections describe in which way the service chaining workflows have been implemented. All workflows are expressed using the workflow language WS-BPEL and are deployed as service instances using a BPEL workflow engine (ActiveBPEL). All workflows are available as BPEL documents in Appendix C.

- **Forest Fire Risk Assessment Service (FFRAS)**: This service chain instance controls the workflow of the processing steps of the forest fire application. It offers three operations (ForestFireFrequency, ForestFireDensity and ForestFireRiskClasses) implementing the three possible analyses in the forest fire application area. The WSDL description of the service is given in Appendix B.1. NOTE: The introduction of the repository service into the chain was necessary because the used Processing Services require data references as inputs, and the FAS cannot provide its output data as such references. This problem is further discussed in Section 5.2, the BPEL document in Appendix C.1
- **Flood Simulation Service (FSS)**: This service chain instance controls the workflow of the processing steps for flood simulation. It performs a flood simulation for different rivers: Rhine, Moselle, Sarre and Neckar. The response is a flood extent coverage (0=no flood 1=flood). This simulation is based on a Grid Coverage of the elevation difference to the closest river and user input of the flood height to be simulated (for a specific area of interest). The flood extent is then calculated using the Map Algebra

[17] Accessible at http://52north.org/index.php?option=com_projects&task=showProject&id=21&Itemid=80

Service's LocalRating operation. The WSDL description of the service is given in Appendix B.2, the BPEL document in Appendix C.2

- **Damage Assessment Service**: This service chain instance controls the workflow of the processing steps for damage assessment. It allows assessing a damage that can be expected from an event with a certain extent. The damage types are fixed and have to be chosen from a list. Currently, only *number of affected people* is supported, which is calculated based on a tightly coupled dataset showing population density. The event extent should lie within the maximum bounding box of the service, which marks the area for which the damage information can be returned. The service chain instance is not fixed to a certain type of event, but works simply on the extent of any event that is considered hazardous. The damage will be calculated for a particular aggregation level. Currently, only NUTS3 can be chosen. The WSDL description of the service is given in Appendix B.3, the BPEL document in Appendix C.3.

4.4. The PEUNHA Client

4.4.1. The Orchestra Architecture Client Components

The set of Orchestra Architecture Client (OAC) components is a library for the development of (client) applications that access Orchestra services.

The OAC runs inside the commonly used web browsers; Mozilla Firefox and Netscape Navigator are currently supported. User interaction starts by loading the HTML main page. The modules on the server side use Apache Tomcat as a servlet engine. The core module is Mapbuilder, which communicates with Orchestra Services or other OGC services by means of the Java Connectors.

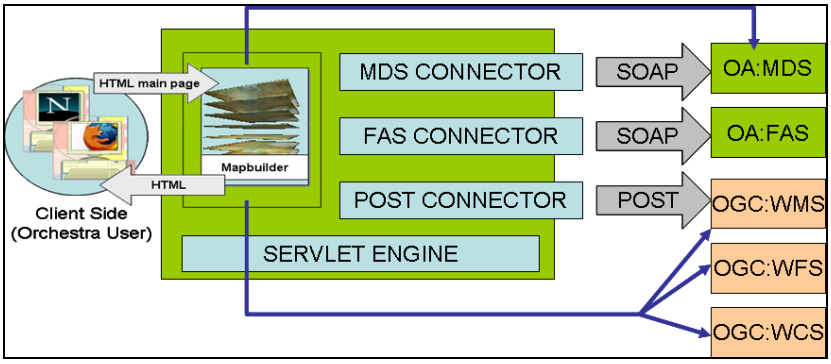


Figure 39: Architecture of the ORCHESTRA Architecture Client

Software requirements for installation are the following:

- A Servlet-Engine (software has been tested on Apache Tomcat 5.5)
- Java JDK 1.5
- Java-Web-Start
- Mozilla Firefox 1.3 ff. or Netscape 6 ff.

The development of the OAC has followed three main phases:

- integration of Mapbuilder functionalities
- development of widgets
- development of Java Connectors for accessing ORCHESTRA services via SOAP.

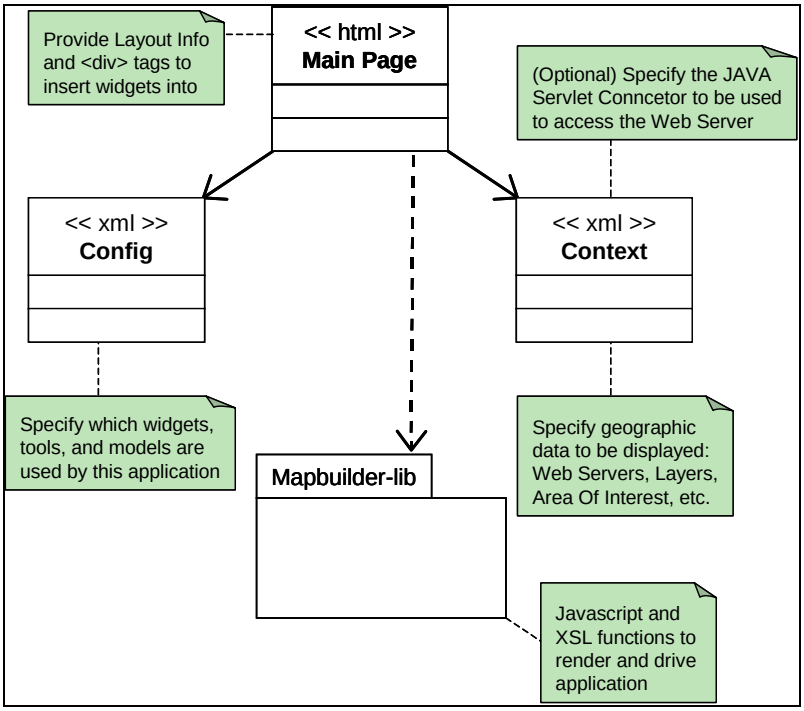


Figure 40: Architecture of a Mapbuilder-based application

Development work has started from the typical architecture of a Mapbuilder-based application. Additionally, for each application (for instance, the Forest Fire Risk Assessment application), the appropriate html Main Page and xml Context and Config files have been created, in order to define the graphic layout of the user interface, and the set of functions to be available via the interface. Context files are developed in compliance with the OGC Web Map Context Documents specifications version 1.0.0 ([15]).

4.4.1.1. Mapbuilder

The OAC is based on *mapbuilder-lib* v 1.0. Version 1.5 is supported, as well. Mapbuilder-lib 1.0 is fully described in the document "*mapbuilder-lib Software Design Description*"[18].

The mapbuilder-lib has been used and extended to implement, within the OAC, all required functionalities for the interaction between the end user and the web services involved in the application, such as map viewing and browsing, showing/hiding thematic layers, geographic data selecting and querying, feature editing, etc.

Mapbuilder supports OGC standards to communicate with external map servers, and provides a set of components which are dedicated to the development of web client applications for GIS. These components can be either widgets or tools. *Widgets* are those *tools* that imply some graphic rendering in the user interface, e.g. an active button. They can be embedded into an application as elements of the user interface (such as a button which activates the zoom tool or a frame for viewing maps), that provide access to specific functions. The underlying idea is that anyone who can handle HTML and JavaScript at a basic level can use a pre-defined widget/tool in newly developed web pages. The set of components is described in the Mapbuilder documentation[19].

The same design approach (i.e. a set of generic, customisable widgets and tools to be incorporated into HTML web pages) has been applied to the OAC development. The OAC is composed by a set of general purpose widgets/tools for accessing the Orchestra services. Further, some pilot application specific widgets/tools have been developed for the implementation of particular functions or interaction steps required by different workflows. Several mapbuilder components have been re-used and integrated; some of them have also been

[18] Available at: http://mapbuilder.sourceforge.net/docs/design/index.html#id2449360

[19] Available at http://docs.codehaus.org/display/MAP/Component+register

modified in order to ease integration with the Orchestra Architecture, and particularly to improve interaction with the Orchestra Map and Diagram Service (MAS) and the Orchestra Feature Access Service (FAS).

In the following section, the main tools and widgets that have been developed for customising the user interface of the pilot applications are described.

4.4.1.2. Orchestra Widgets & Tools

OA_Map_Pane and Button Bar: The main map pane and the button bar are described together, since they are closely interacting with each other.

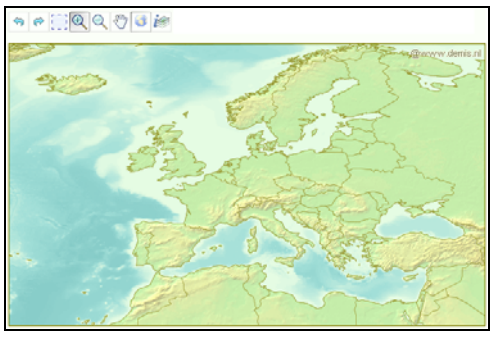


Figure 41: Main map pane and button bar

The main map pane is used to show the current map; by means of the tools in the button bar, users can do the following (buttons from left to right on the pane, see figure above):

- switch back and forth between two different visualisations (**back** and **forward** buttons);
- select an area of interest (see below, **OA_SetAOI** and **OA_SetDropAOI**) that will be taken into account for risk analysis purposes;
- zoom in and out the map (**zoomIn** and **zoomOut** buttons);
- pan the map (**dragPan** button);
- get back to the full extent visualisation (**reset** button);
- get information about geographic features (**getFeatureInfo** button).

Further, the OA_Map_Pane widget accomplishes the task of building and submitting requests to the Processing Client (see description below), by concatenating the various parameter values set by users in the main WEB page of the application.

The background layer(s) of the map can be selected by the user via the application settings WEB page. New layers can be added either from an OGC WMS or an OGC WFS, or from an ORCHESTRA MAS or FAS (cf. Figure 42).

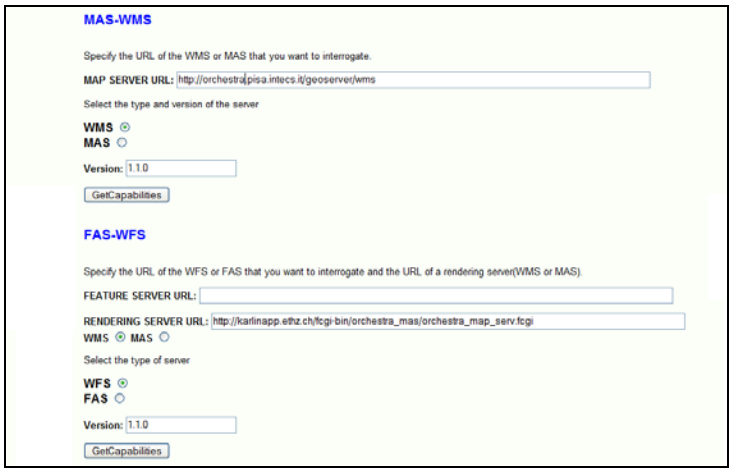


Figure 42: Applications settings page - adding new layers

In order to load WFS/WMS layer, the user can optionally submit a GetCapabilities request, and eventually select layers (and the associated style descriptor) to be added to the map by clicking in the corresponding check-box(es) (cf. Figure 43).

Layer attributes	Add	Style
Layer title: EurCapitals_Type **Layer name:** topp:EurCapitals **Layer coordinates:** EPSG:4326 **Layer Bounding Box:** minx:-21.920780000000377; miny:35.16870000000017; maxx:33.371030000000246; maxy:64.14372999999978;	☐	eur_capcities_style.sld
Layer title: RhineEurogrid100Impact_Type **Layer name:** topp:RhineEurogrid100Impact **Layer coordinates:** EPSG:4326 **Layer Bounding Box:** minx:2.3010641742405515; miny:46.29957310203877; maxx:11.171384510041165; maxy:52.80841557673825;	☐	eur_capcities_style.sld
Layer title: giant_polygon_Type **Layer name:** tiger:giant_polygon **Layer coordinates:** EPSG:4326 **Layer Bounding Box:** minx:-180.0; miny:-90.0; maxx:180.0; maxy:90.0;	☐	eur_capcities_style.sld

Figure 43: Application settings page - select layers to be added

Other functionalities are:

OA_WFSGetFeature: This tool is a modified version of WFS_GetFeature; it has been created to interact with OA_LayersQuerySelection.

OA_LayersQuerySelection: This widget allows the user to define what map layers are sensible to the GetInfo tool. It affects the functioning of the widget OA_SetAOI.

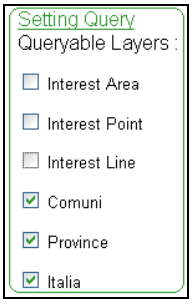


Figure 44: OA_LayersQuerySelection

OA_RadioContext: This widget allows the user to choose among different layers to be displayed, within the same area of visualisation. This means that, according to the user's selection, the HTML page is loaded or refreshed according to the appropriate Context file. This widget refers to a configuration file which can be modified by means of the configuration page of the client application.

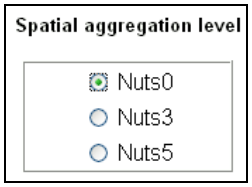


Figure 45: OA_RadioContext

Also this widget can be customised by the user through the application settings WEB page, as in the case of the OA_Map_Pane widget (see above). In this case, the user will choose a layer to be used for each spatial aggregation level.

OA_SetAOI: This widget activates the selection tool on the main map pane. The Area of Interest (AOI) in form of a box is drawn on the map starting from the upper left corner. The user can either set the box by pointing and dragging the mouse, or inserting latitude and longitude values of the tie point in the pop-up window.

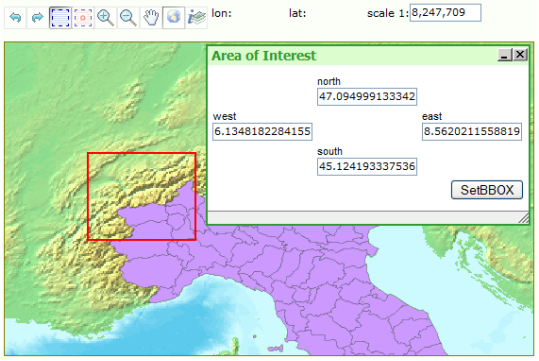


Figure 46: OA_SetAOI

OA_SetDropAOI: This widget activates the selection tool on the main map pane. The AOI is drawn on the map starting from the centre. The user can enter the AOI bounding coordinates values in the text-boxes of the pop-up window. Alternatively, the user can define X and Y resolution of the AOI in map units, the AOI width and height in pixel and then define the centre of the area; such tie-point can be either defined by pointing with the mouse on the map, or by setting the point coordinates values in the text-boxes.

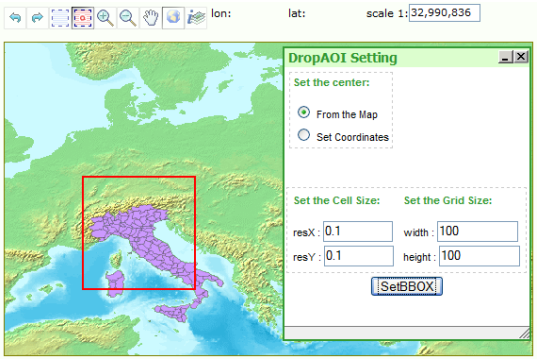


Figure 47: OA_SetDropAOI

OA_Header: This widget allows customising the header, the title and the logo (along with its link) to be displayed on the client web pages.

Figure 48: OA_Header

OA_ClassLegend: This widget creates and shows the legend for a map; for each layer in the map the <Title> of the corresponding <Rule> in the StyledLayerDescriptor is displayed. The symbolisation can be modified interactively by users through the Edit Symbolisation pop-up window in the result page of the application.

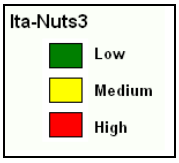


Figure 49: OA_ClassLegend

OA_RangeData: This widget prompts the user with input boxes to enter dates in a controlled manner. It is useful to generate queries specifying a date value range (for instance the temporal extent of analysis in the forest fire application). Entered dates are checked for consistency.

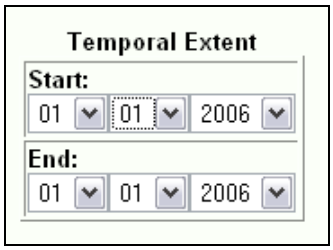


Figure 50: OA_RangeData

4.4.1.3. Java Connectors

A major part of the OAC is constituted by the three Java Connectors which handle interaction among the Mapbuilder module and the Orchestra services. These components address the problem of allowing different communication protocols in data exchanges among the client application and the external services. This need stems from two main considerations:

- SOAP communication, which is not supported natively by the Mapbuilder library, is required by the Orchestra Project specifications for Web services.[20]
- HTTP GET and POST communications are alternatively supported by Mapbuilder for different cases of interaction. However, more flexibility was needed by the OAC (particularly, in order to use HTTP POST communication in some cases that required HTTP GET communication).

The Java Connectors appeared to be a flexible solution towards full interoperability; at the same time, the option of using standard OGC services (particularly, the WMS) in the development and test phases has allowed integration of a number of operations before they were available (also) on Orchestra services.

MAS Connector and FAS Connector. These two modules are Java servlets which manage SOAP interaction between the mapbuilder module and the Orchestra MAS.

POST Connector. The POST Connector is a Java servlet, which has been developed in order to use the mapbuilder widget “MapPane” without major modifications. MapPane, which is one of the main components of Mapbuilder, is in fact based on the HTTP GET communication, which makes complex requests hard to handle.

[20] See the project deliverable D3.2.3 “Reference Model for the ORCHESTRA Architecture”,

Particularly, serious problems have occurred when trying to append Styled Layer Descriptor (SLD) statements to the requests. This connector, although being suitable for different cases of application, has been initially developed to manage interaction with the WMS. Using the WMS made it possible to develop the OAC (and also the Classification Service) while the MAS was still under development by enabling functionalities such as map legends, image transparency, or OGC Filter encoding. A POST connector for the WFS was not required. The POST protocol is required for editing functions only, and Mapbuilder already gives support to WFS-Transactional requests for editing operations.

These connectors generate an additional computation step in data exchange among the components. However, this extra-processing does not affect the system performance: the connectors run locally with respect to the client application, and their processing time is negligible with respect to the time spent in data exchanges with the services. Further, these connectors avoid interfacing other components of the client application (namely, JavaScript-based components) with a proxy application, which would be on the contrary needed to communicate with external services.

4.4.2. The Processing Client

The main architectural difference with respect to the OAC is given by the Processing Client servlet, which manages user's input and builds requests to the needed data and/or processing services (in Figure 51, only a subset of requested operations are listed).

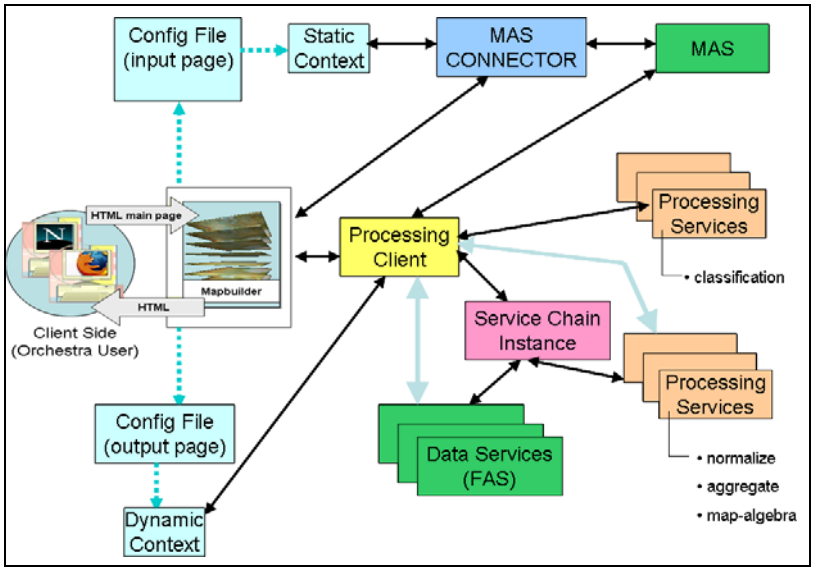


Figure 51: Architecture of the PEUNHA Client

The Processing Client controls the execution of the workflow describing the thematic applications: it manages requests generated through the WEB pages of the client application and submits them, in the appropriate format, to the external services (service chain instances, Map and Diagram Service etc). Furthermore it manages the data caching mechanism within the client application by saving locally the data that is returned by Processing Service instances to the Client application (for instance, a SLD generated by the classification service).

The Processing Client accomplishes the following tasks:

- collect and concatenate the user's input parameters into one request (type of statistics, aggregation level and time range);
- build and submit requests to the service chain instance;
- build and submit request to the Classification service;
- send Classification service result (Styled Layer Descriptor) to the map server in order to have it rendered);

- handle the data caching mechanism.

4.4.3. PEUNHA Client WEB Pages

The PEUNHA client interface is composed by two main WEB pages; each page is managed by a specific Java Script:

- The **Input page** (or main page) is mainly used to allow users specifying the parameters for data processing and analysis (see Figure 54 and Figure 56).
- The **Output Page** shows users the processing results. Users can change the map symbolisation options (see Figure 55 and Figure 60).

Other accessory pages are:

- **Application Settings:** In this page users must set the servlet container "home", the name of the service and the path for cached data.

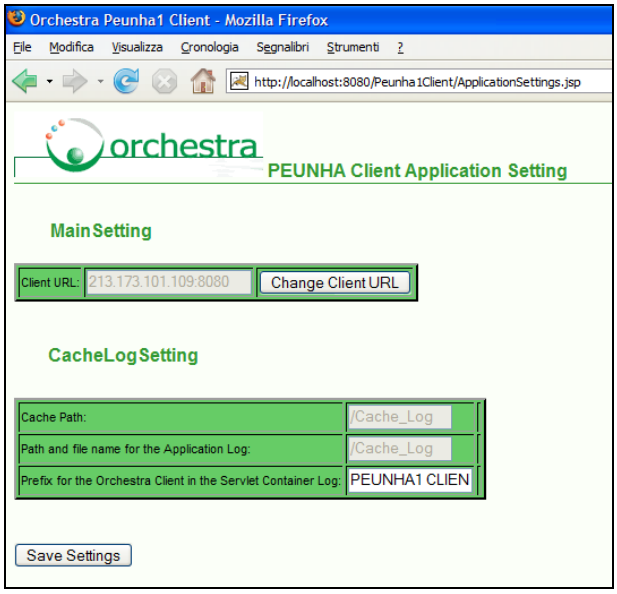


Figure 52: Application settings

- **Edit Symbolization**: This is a pop-up window in the output page of the application (Figure 55). In this window, the user can change the symbolisation options for classified maps.

4.4.4. PEUNHA Client User Interface

A dedicated client application has been developed for the pilot. This application handles user interaction according to two different workflows, corresponding to the Fire Hazard Assessment use case and to the Flood Simulation and Damage Assessment use case.

The two different workflows are accessed from the main page of the PEUNHA Client (Figure 53):

Figure 53: Screen shot of the PEUNHA Client main page.

4.4.5. Forest Fire Risk Assessment Client

In the forest fire application, the user can choose the kind of analysis, the level of spatial and temporal aggregation, the temporal and spatial extent (see Figure 54). Possible analyses include the frequency (selected in Figure 54), the density, and the risk of forest fires. Spatial aggregation may follow administrative boundaries at different levels (NUTS). The spatial extent can be selected as a bounding box by the user (shown in red in Figure 54). The temporal extent can also be selected.

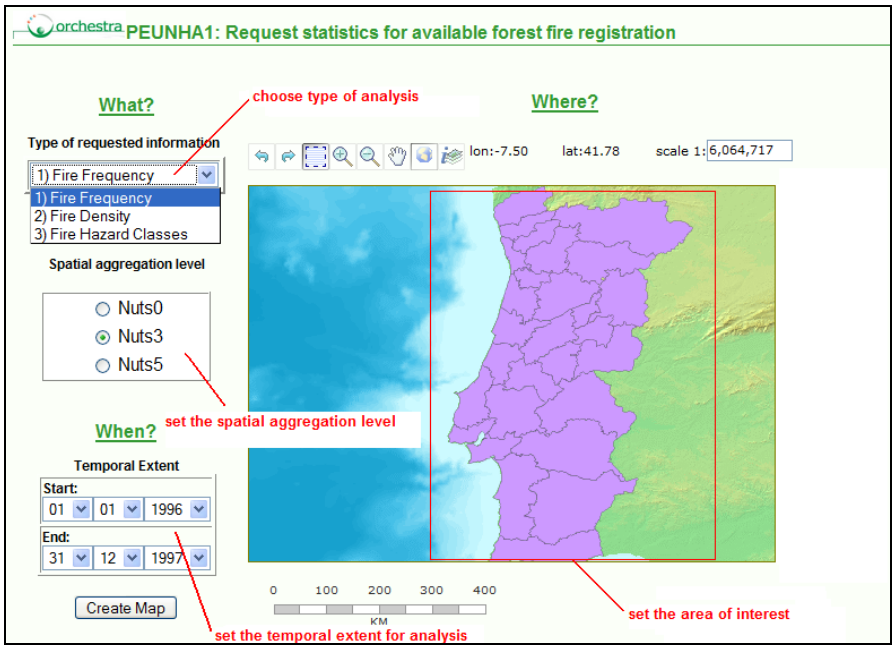


Figure 54: Screen shot of the client interface – Request forest fire analysis.

After the analysis, the client application shows the results in a map (Figure 55) using a pre-defined symbology. This symbology can be changed by the user (as shown in the box in Figure 55). The user can select a different map extent by zooming in/out or panning, or select features in order to visualise their attributes.

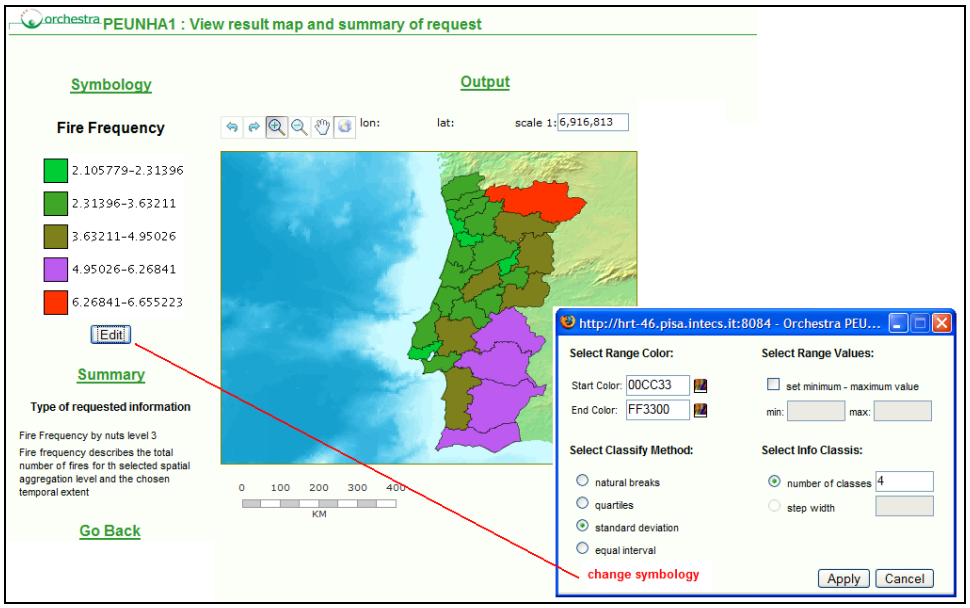


Figure 55: Screen shot of the client interface – presentation of the results.

4.4.6. Flood-related Damage Assessment Client

The flood-related damage assessment application is built up of two components: the user can access either the Flood-extent Simulator application which interacts with the Flood Simulation Service (FSS), or the Damage Assessment Calculator application (Figure 56) which interacts with the Damage Assessment Service (DAS).

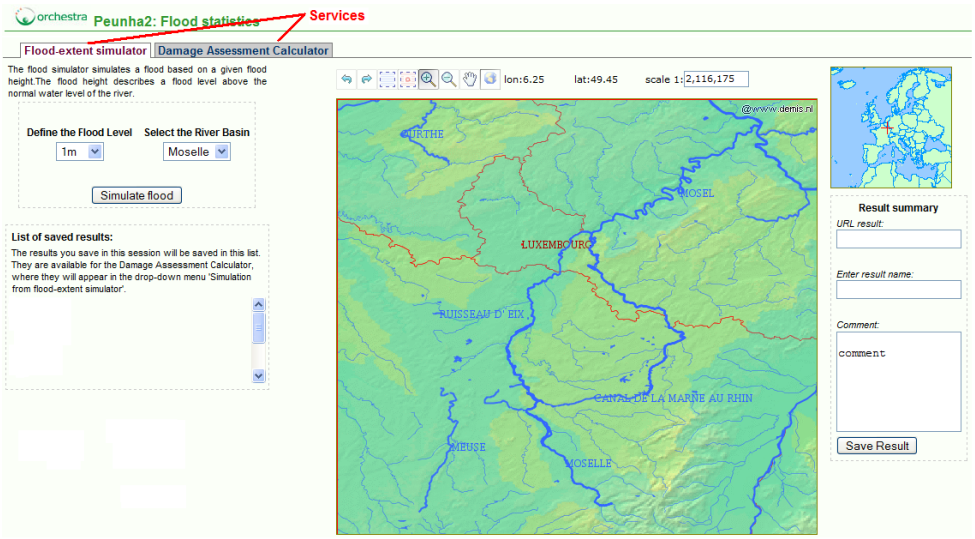


Figure 56: The main page of the Flood-related Damage Assessment Application, with its two components

The FSS aims at creating hypothetical scenarios of flooding events. The user has to select a river basin and a flood level. An area of interest can be defined, as well (Figure 57).

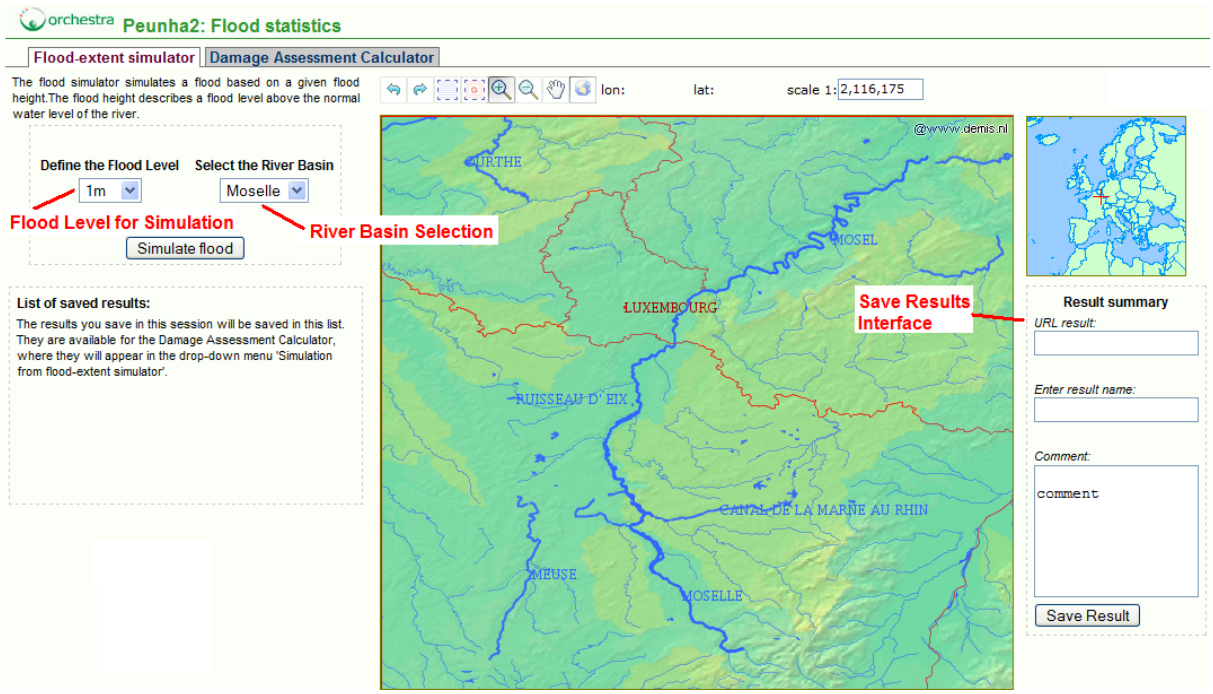


Figure 57: Flood-extend simulator which interacts with the Flood Simulation Service

After the processing phase, the user can save results, which are kept available as far as the HTTP session is active. Resulting datasets are saved in GeoTIFF format (Figure 58).

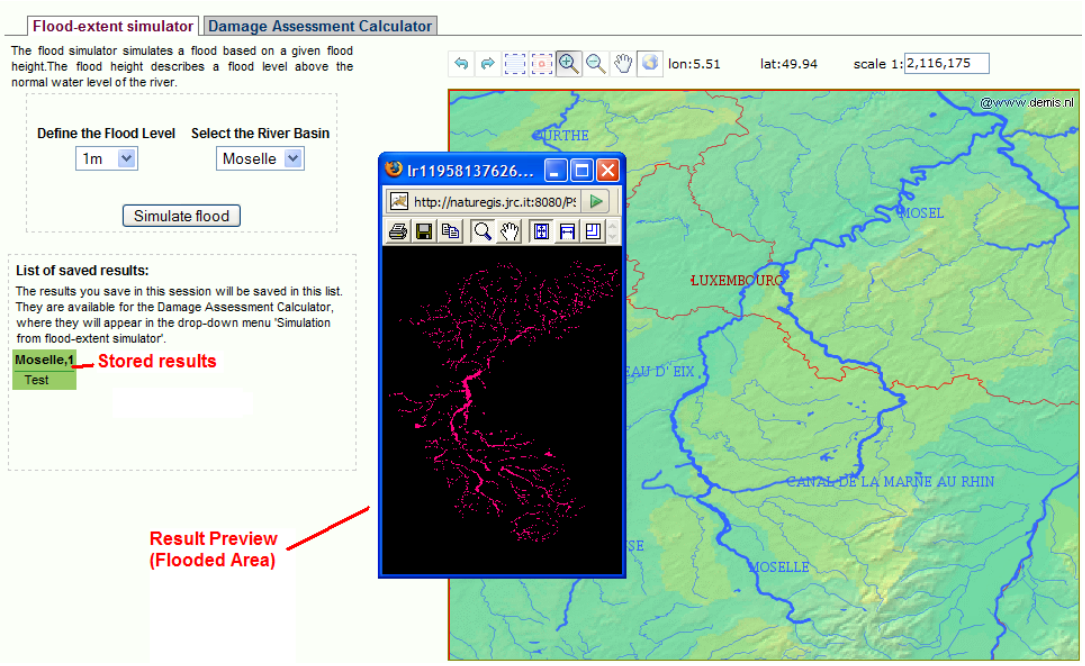


Figure 58: Flood Simulation Results

In the DAS page, the user can choose to submit as Flood-extent parameter either historical data or FSS results. Furthermore, the spatial aggregation level and the damage indicator can be specified.

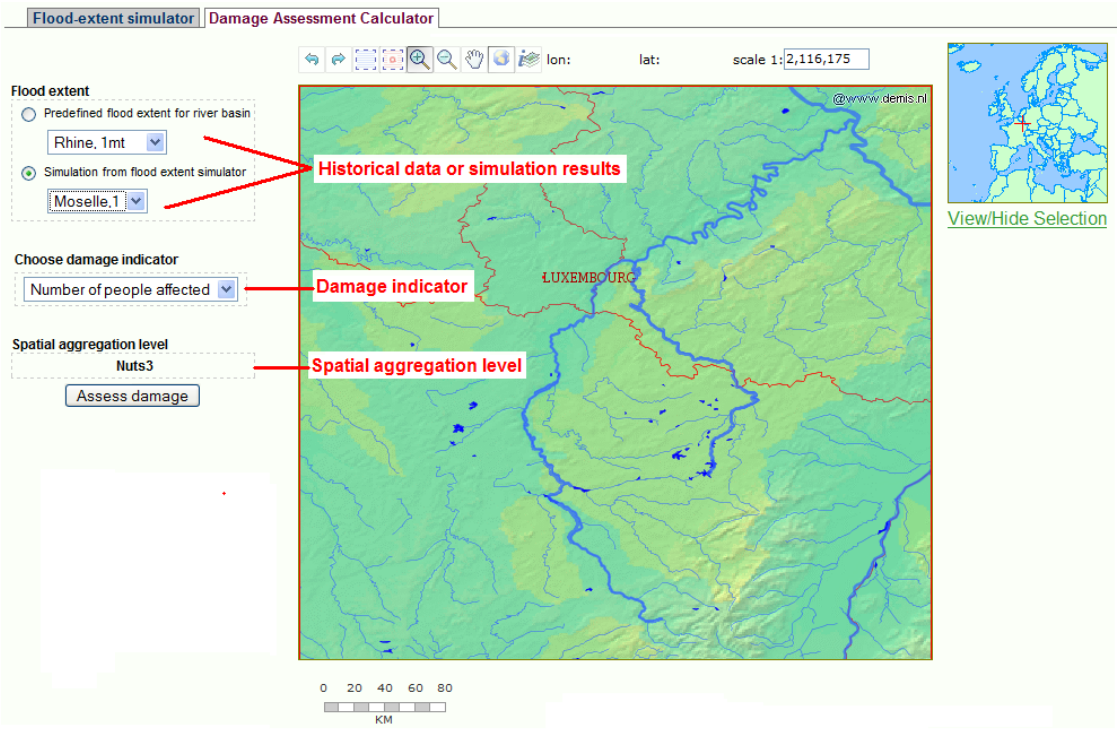


Figure 59: Damage Assessment Calculator which interacts with the Damage Assessment Service

The application shows the results in a map using a pre-defined symbology. This symbology can be changed by the user (as shown in Figure 55). The user can select a different map extent by zooming in/out or panning.

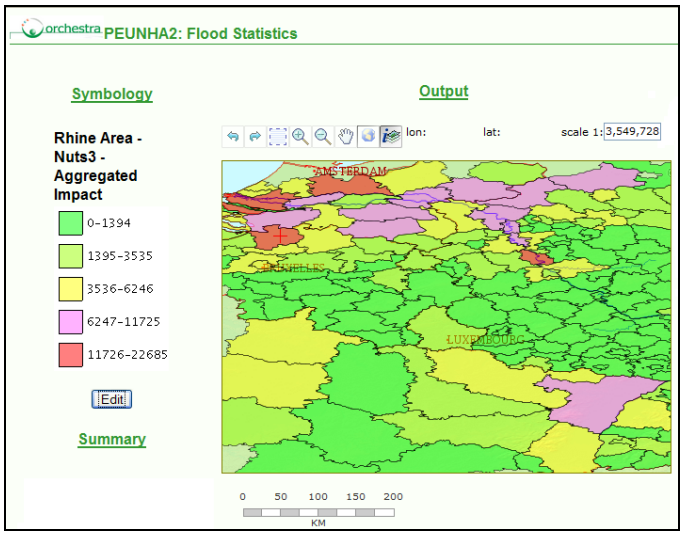


Figure 60: Damage Assessment Result Page

5. Lessons learnt & Recommendations

The scope of the ORCHESTRA pilots was to test the RM-OA in real-world case studies. Throughout the implementation we came across various issues that range from general architecture-issues and SOA-design issues to specific encoding of messages and data types. In this section we summarise abd present some of these issues, the solutions we applied and the recommendations to the GI community we derive from them. We use the following structure:

Issue	
Problem:	*A brief description of the problem that was encountered during the pilot implementation. The problem might refer e.g. to one or both of the presented pilot applications or a specific service type such as the Feature Access Service.*
Solution:	*The solution that was applied for the problem during the pilot implementation. A problem might remain unresolved.*
Recommendation:	*A recommendation derived from the problem and the applied solution, as well as any related research that was carried out by the JRC-IES ORCHESTRA team during the ORCHESTRA project. The recommendation is meant to generalize from the pilot application and the chosen solution to be applicable in other, similar SOAs and SOA based applications for risk analysis and damage assessment.*

The issues have been organised into three different sections, from the general to the specific: Architecture and design patterns, Interface design and Message and Data encoding.

5.1. Architecture and design pattern

This section is of general nature. It contains issues related to the architecture of the pilot applications, and, derived from that, the improvement of any SOA for the hazard, risk and damage assessment.

Translating Feature Access Service & the scenario of MS proving their data in a common schema

Problem: A main lesson from the forest fire application area is that the heterogeneity of basic data provided by member states (resulting from different languages, schemata, CRS and data formats used) presents a serious obstacle for pan-European hazard and risk analysis and mapping. This is true even though the implementation rules to the Forest Focus Regulation (EC No 2152/2003) describe the information to be provided.

Solution: In the pilot, this issue has been addressed by using a FAS-X. The FAS-X provides access to harmonised member state information through standardised interfaces. This service maps from local (i.e. member state) schemas to a common schema that is based on the information required by the implementation rules to the Forest Focus Regulation. Mapping rules from each local schema to this common schema had to be developed, which proved to be difficult without having expert knowledge on the local schemas.

Recommendation: Making FAS-X available in the different member states would greatly facilitate the task of performing pan-European analyses. Furthermore, data requirements should, if possible, not only be expressed in plain text (as the above name Forest Focus Regulation), but also as a data model (e.g. in UML) and/or encoding (e.g. in XML). The mapping from the local schemas to a common schema should be performed by domain experts in collaboration with the data providers, in order to prevent errors.

Hiding details from users – not too much detail

Problem: To raise awareness among stakeholders is an important task for decision makers dealing with risk reduction. Yet, decision makers are concerned about (1) misuse of detailed real-world or forecasted data on hazards and related damage by third parties and (2) about false concerns this might raise among citizens who are unaware of aspects of data quality, reliability and spatial resolution. This particular concern was expressed by the decision makers involved in the specification of the flood simulation & damage assessment application.

Solution: In the pilot this issue has been addressed by means of opaque service chaining: While the analysis of the damage resulting from either a simulated flood or from historic data is performed on a detailed grid of 100x100m, the results of this analysis are then aggregated by administrative areas. By choosing opaque chaining, the end user of this application, i.e. the citizen, has no access to the underlying business logic and thus no insight in the intermediate results of the detailed analysis.

Recommendation: When dealing with sensitive data and workflows, using opaque chaining to hide the underlying business logic is a sensible approach: only those inputs and outputs are revealed to the user, which the decision maker considers suitable. The business logic, containing possibly sensitive workflows and data sources, and the resulting intermediate outputs remain hidden.

The usefulness of distributed geo-processing for the given applications

Problem: In order to guarantee a sustainable pilot development (particularly an easy development of new risk analysis) the pilot is designed for supporting a flexible combination of geo-processing functionalities and a case-based visualisation of results as a base-paradigm.

Solution: Workflows are particularly suitable for expressing how to combine available services. Thanks to workflow engines applications can delegate to them the control of service chain executions. Flexibility is guaranteed by the fact that the service chains can use any available geo-processing services applied to any data available from either other services or user inputs without implying any changes at the application using it. This pilot contributes to this challenge through providing access to standardised interfaces for processing and visualisation of spatial data. They are combined into value-added service chains, whose creation is supported through the use of semantics and ontologies facilitating the identification of the most appropriate processing functionalities and data.

Recommendation: The process of creating new information shall be decoupled as mush as possible from the application needing that information. This can be achieved by using workflows and workflow engines allowing minimizing the impact on the system configuration when new relevant geo-processing and data are integrated.

Passing data by value versus passing by reference

Problem: Distributed geo-processing requires the transfer of intermediary results from one service to the next. The larger the dataset is the greater is the impact on the performance of the distributed processing.

Solution: The introduction of the data-passing modality (i.e. passing data by value or by reference) in service parameters allows us to avoid redundant data transfer. This happens especially when the data is passing through the central workflow coordinator because most of the time it does not need it. The solution is to decouple the workflow in control flow and data flow so that this concern can be adequately faced in the data flow study. For further details and results about this approach see (Friis-Christensen et. al 2009).

Recommendation: When designing workflows consider the (explicit or implicit) corresponding control and data flow for identifying potentially unnecessary data transfers that should be replaced with a reference at the data.

Exposing tools of existing GIS software packages as PS operations

Problem: The pilot has shown the feasibility of exposing functionalities of existing GIS software packages (e.g. GRASS) as a processing service. However, while GRASS is widely-used and reliable, it is not based on a client-server architecture, which has lead to a number of problems and workarounds during the implementation of the GRASS-based processing services.

Solution: In the pilot, a dedicated server application has been developed, which handles requests and generates scripts to the GRASS engine, allowing concurrent access.

Recommendation: The benefits of using well-established GIS software as the basis for developing geoprocessing services should be weighed against the drawbacks of having to develop workarounds if the GIS package is not available as a server product or API. There are already initiatives that address these problems, e.g. the pyWPS (http://pywps.wald.intevation.org/) WPS implementation for GRASS. Unfortunately, when the pilot development started, this implementation was not yet available.

Asynchronous control pattern

Problem: We experienced several performance-related issues: Firstly, when executing a workflow, the computational time required could be very long, particularly when the workflow was complex. Secondly, the time required to parse and validate a GML data stream that was used as an input into the workflow could be lengthy. Finally, the transport of the data itself might impact the performance.

Solution: An asynchronous control pattern was used in order to prevent timeouts and unacceptable waiting time for the users. Notification mechanisms based on e-mail informing the user about the availability of the result and how to visualize it in the application have been developed.

Recommendation: In a distributed context the immediate availability of processing results cannot be guaranteed, therefore it is worth evaluate the introduction of separate funcionalities for submitting a request (e.g. risk analysis) and for getting the results.

5.2. Interface design

Returning Data by Reference

Problem: The Feature Access Service (FAS) is, according to its specification, only able to return features by value, i.e. as a GML document or as a coverage (which is then encoded in a string). In contrast, the Processing Service instances used in the service chains in the JRC-IES pilot, require input data to be passed by (URL) reference.

Solution: A Repository Service was introduced, which makes results of a FAS request available as a URL. Two types of this service are used in the pilot, one for storing GML data, and one for storing GeoTIFF. Both types receive a string (containing the GML or the base64-encoded GeoTIFF) as input, store the data locally (in the case of GeoTIFF in the binary format) and make it accessible through a URL, which is returned as a result. The solution of using two types of Repository Service (rather than one that includes the format as an additional input parameter)

was chosen for simplicity.

Recommendation: An (optional) attribute could be added to the GetFeatures operation in the FAS interface. This attribute would allow the request to specify whether the features should be returned by data or by value, i.e., whether a GML stream or coverage or a URL should be returned. This would mean that the FAS would have to implement the functionality of the Repository Service. In case store is set to false and data is passed by value in the response, the way how it is transmitted could imply the encoding: If it passed in the SOAP body, it is passed using base64 encoding, but if a SOAP attachment is passed the original binary format can be used.

Spatiotemporal Domain of Service Chain Instances

Problem: All service chain instances are based on particular data sources. But as these sources are unknown to the user, it is well possible to pose requests to the service that do not make sense, e.g. using a bounding box that is outside the spatial coverage of the used data.

Solution: In the pilot, the client application and the service chain instance are very tightly coupled. This ensures that only sensible requests are being sent to the Service Chain instances in terms of the bounding boxes.

Recommendation: A tight coupling of client application and service chain instance is not always a suitable solution. Therefore we strongly recommend creating service chain instance meta-information that contains, among other, the spatiotemporal domain of the service.

PS/WPS interface and message encoding and the overhead when chaining

Problem: The use of several PS execute requests as part of a possibly lengthy service chain using BPEL is error-prone and tedious. A major problem is, that it is not possible to validate a process-specific execute request, beyond its compliance with the PS execute request schema.

Solution: This problem has not been solved in the pilot.

Recommendation: Integrating web services into a BPEL process that have a WSDL description that immediately focuses on the relevant service inputs and outputs, without the detour of being wrapped into an execute request, might be the more simple solution. The WPS specification, which underlies the PS Specification, foresees this through an optional specification of WSDL for an individual WPS process.

Coordinate Reference Systems

Problem: Some services have constraints on the CRS of the consumed data. For example, the normalization Processing Service used in the forest fire example requires coordinates to be in an area-true CRS, such as ETRS89-LAEA (EPSG:3035), in order to calculate a correct result.

Solution: No coordinate transformation service was used in the pilot implementation, Rather, all coordinates were assumed to be in the same CRS: the widely supported WGS84 (EPSG:4326). For the normalization by area operation, a functionality was included that reads the given coordinate transformation for a feature collection and, if necessary, transforms the data into an area true reference system (ETRS89-LAEA in our case) and then calculates the true area values. The geometry of the return data is still kept in the original coordinate reference system. This approach was preferred to the (more flexible) option of having a separate service responsible for coordinate transformation for performance reasons.

Recommendation: Based on relevant user requirements, it should be decided early on whether coordinates in a given service chain are to be restricted to one or several (few) CRS. Depending on the number of CRS used and performance requirements, a decision should be also taken whether to include a service for coordinate transformations in the architecture or to do the required transformations behind the scenes.

5.3. Message and Data Encoding

Character Encoding

Problem: The FAS and FAS-X services retrieve a feature collection from a Web Feature Server through a HTTP request. The HTTP response, a byte stream, must be converted to a character stream before the XML document containing the feature collection can be processed. In order to perform this step the character encoding used must be identified.

The character encoding of the returned XML document cannot always be easily determined. The “charset” parameter in the HTTP Content-Type header might be missing or, if present, it might be inconsistent with the value of the “encoding” attribute of the XML document prolog. In addition, when dealing with character sets like UTF-16 and heterogeneous hardware and software platforms, the endianness might also introduce conversion issues. XML parsers (e.g. Woodstox) are able to probe the stream in order to establish the character encoding used and they check that it corresponds to the one declared in the xml prolog. The java utilities (java.io package) that convert a byte stream into a character stream are XML-unaware and cannot therefore be fully trusted when dealing with XML documents.

Solution: Two different approaches have been implemented:

- In the FAS-X the HTTP stream coming from the WFS is converted to a java string explicitly specifying the encoding detected from the HTTP response header.
- In the FAS the HTTP stream is instead directly passed to the Axis2 STAXOMBuilder which, through a low level parsing, detects the physical character encoding scheme.

Recommendation: Define a common encoding or a common mean for expressing the used encoding.

Feature Encoding

Problem: Once the XML document has been reconstructed, the feature collection must be returned in the SOAP envelope as required by the FAS. The choice of returning the feature collection in an element of type XML Schema “string” implies that the feature collection undergoes an additional encoding since the original XML document must be transformed into parsed character data. This encoding bloats the feature collection even more than it already is.

A different solution is to include the feature collection in an element whose type definition is XML Schema “anyType”. This type is generic allowing both scalar and non-scalar data. A particle (xsd:any) would give a better representation since it allows only for non-scalar data. However the particle is not a type and its representation in UML is clumsy.

Solution: Two different approaches have been implemented:

The FAS-X returns the feature collection in an element of type XML Schema “string”. However, in the Java implementation, the handling of String objects needs the entire stream to be fully acquired from the WFS before the SOAP

envelope can be built.

In the FAS, the following approach has proved to be more efficient, in terms of speed and occupied memory: The feature collection is returned in an element of type XML Schema "anyType". This type is generic allowing both scalar and non-scalar data.This has made possible the following strategy: the HTTP stream is fed to a STAX (Streaming API for XML Processing) node builder. The resulting node tree is directly attached to the corresponding element of the SOAP envelope. The node builder has a capability of deferred parsing. This means that the http stream is consumed while the envelope is built and sent. This avoids storing the feature collection locally.

Recommendation: Instead of sending the feature collection as a plain XML fragment, it could be encoded according to the Fast Infoset[21] standard.

GeoTIFF Raster Shift

Problem: A problem has been detected in the import function of GeoTIFF raster files when they are interpreted by the PS Map Algebra Service. The GDAL library, which is used in for the implementation of the GeoTIFF import, does not handle the Metatag "AREA_OR_POINT" in the GeoTIFF header, which describes the point of origin to be used for generating the raster. This might result in a spatial shift (typically by half a pixel). This problem has been revealed using a number of tests which are described in detail in Appendix D.

Solution: The issue has not been solved in the pilot.

Recommendation: The problem has to be solved by the GeoTIFF/Grid Coverage community and the applications GRASS and GeoServer.

The limitations of GeoTIFF in complex service chains

Problem: In the flood damage assessment application, one intermediate output of the service chain is the number of people affected by the flood, aggregated for each administrative unit (i.e. NUTS region). The output is in a tabular format and supposed to be joined to the NUTS features by using the NUTS code (the identifier of the NUTS areas) as the join attribute. This table is an output of the ZonalSum operation of the Map Algebra Service. Input to the ZonalSum operation are a Grid showing the population density and a Grid showing the NUTS regions. However, the identifier of the NUTS regions is a String (the NUTS Code), consisting of a combination of letters and numbers. Thus, representing the NUTS Code in a GeoTIFF is not possible, as the GeoTIFF Format, like most binary raster formats, only allows for the representation of numeric values.

Solution: In the pilot study we were bound to the use of GeoTIFF: as a de-facto standard for the representation of raster data it was supported by the chosen software solutions that were underlying the FAS and the Map Algebra Service. Thus, as a workaround a unique numeric identifier has been added to the NUTS data to represent the NUTS code. This attribute is used to join the output of the Map Algebra Service to the NUTS feature.

Recommendation: The chosen solution is a workaround. Thus, in order to overcome the limitations of GeoTIFF, an alternative coverage representation should be considered (such as GML 3.2) that allows handling of String values.

[21] http://asn1.elibel.tm.fr/xml/finf.htm

References

1. ORCHESTRA, *ORCHESTRA. An open service architecture for risk management*. 2008: The ORCHESTRA Consortium.
2. Erl, T., *Service-oriented Architecture: Concepts, Technology, and Design*. 2005, Upper Saddle River: Prentice Hall PTR.
3. ORCHESTRA, *Reference Model for the ORCHESTRA Architecture (RM-OA) V2 (Rev 2.1)*, in *OGC Best Practices*, Open Geospatial Consortium, Editor. 2007.
4. Cockburn, A., *Writing effective use cases*. 2001, New York: Addison-Wesley.
5. Fowler, M., *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Object Technology Series, ed. G. Booch, et al. 2003, Boston, San Francisco, New York, Toronto, Montreal, London, Munich, Paris, Madrid, Capetown, Sydney, Tokio, Singapore, Mexico City: Addison Wesley
6. Larman, C., *Applying UML and patterns: An introduction to object-oriented programming and design and iterative development*. 3 ed. 2005: Pearson Education Inc.
7. ISO, *Geogaphic information - Services*. 2005, International Organization for Standardization.
8. Barredo, J.I., A.d. Roo, and C. Lavalle. *Continental scale flood risk assessment in Europe*. in *XXIII Conference of the Danubian Countries on the Hydrologic; Forecasting and Hydrological Bases of Water Management.* . 2006.
9. DG-JRC, *European Forest Fire Information System.* 2007.
10. Tomlin, C.D., *Geographic information systems and cartographic modelling*. 1990, New Jersey: Prentice Hall.
11. ORCHESTRA, *ORCH-ImplServ, WP3.6 OA Service Implementation Specifications. Deliverables D3.6.x. Inte-grated Project 511678 ORCHESTRA, Editor: Environmental Informatics Group (EIG)*. 2007.
12. Friis-Christensen, A., et al., *Service Chaining Architectures for Implementing Distributed Geoprocessing Applications.* Journal of Geographical Information Science, accepted.
13. GRASS Development Team. *Geographic Resources Analysis Support System*. 2008 [cited 2008-08-28]; Available from: http://grass.osgeo.org/.
14. M. Hugentobler, I. Iosifescu-Enescu, and L. Hurni. *A Design Concept for Implementing Interoperable Cartographic Services based on reusable GIS Components*. in *XXIII International Cartographic Conference, Moscow, Russia (4-10 August Moscow 2007).* 2007.
15. OGC, *Web Map Context Documents (WMC)*. 2003, OGC (Open GIS Consortium Inc.).

Appendix A: Processing Services Schemas

In the following sections, the DescribeProcess XML documents for the various operations are reported.
Deviations between the conceptual design of the operations and actual implementation are discussed.

A.1 Processing Services for Join, PointInPolygonAggregation and Normalization

A.1.1 Join Operation

```
<wps:ProcessDescriptions xmlns:wps="http://www.opengeospatial.net/wps"><wps:ProcessDescription processVersion="1" storeSupported="false"
statusSupported="false" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <ows:Identifier>org.n52.wps.server.algorithm.InnerJoinAlgorithm</ows:Identifier>
    <ows:Title>Inner join of two GML datasets</ows:Title>
    <ows:Abstract>This operation performs an inner join or intersect of two GML datasets on the basis of two corresponding attribute values.
    </ows:Abstract>
    <ows:Metadata xlink:title="join"/>
    <wps:DataInputs>
      <wps:Input>
        <ows:Identifier>LeftFeatures</ows:Identifier>
        <ows:Title>Features as GML data or reference</ows:Title>
        <ows:Abstract>Features as GML data or reference</ows:Abstract>
        <wps:ComplexData/>
        <wps:MinimumOccurs>1</wps:MinimumOccurs>
      </wps:Input>
      <wps:Input>
        <ows:Identifier>RightFeatures</ows:Identifier>
        <ows:Title>Features as GML data or reference</ows:Title>
        <ows:Abstract>Features as GML data or reference</ows:Abstract>
        <wps:ComplexData/>
        <wps:MinimumOccurs>1</wps:MinimumOccurs>
      </wps:Input>
      <wps:Input>
        <ows:Identifier>JoinAttributeLeftCollection</ows:Identifier>
        <ows:Title>Join attribute of the left Feature Collection</ows:Title>
        <ows:Abstract>Join attribute of the left Feature Collection</ows:Abstract>
        <wps:LiteralData>
          <ows:DataType ows:reference="xs:string"/>
          <ows:AllowedValues>
            <ows:Value/>
          </ows:AllowedValues>
        </wps:LiteralData>
        <wps:MinimumOccurs>1</wps:MinimumOccurs>
      </wps:Input>
      <wps:Input>
        <ows:Identifier>JoinAttributeRightCollection</ows:Identifier>
        <ows:Title>Join attribute of the right Feature Collection</ows:Title>
        <ows:Abstract>Join attribute of the right Feature Collection</ows:Abstract>
        <wps:LiteralData>
          <ows:DataType ows:reference="xs:string"/>
          <ows:AllowedValues>
            <ows:Value/>
          </ows:AllowedValues>
        </wps:LiteralData>
        <wps:MinimumOccurs>1</wps:MinimumOccurs>
      </wps:Input>
    </wps:DataInputs>
    <wps:ProcessOutputs>
      <wps:Output>
        <ows:Identifier>URL_To_GMLResult</ows:Identifier>
        <ows:Title>URL to the Result file</ows:Title>
        <ows:Abstract>The result is a GML file that cointains a feature collection taht contains only those features that are present in both the left and
the right feature collection</ows:Abstract>
        <wps:LiteralOutput>
          <ows:DataType ows:reference="xs:string"/>
          <ows:AllowedValues>
            <ows:Value/>
          </ows:AllowedValues>
        </wps:LiteralOutput>
      </wps:Output>
    </wps:ProcessOutputs>
  </wps:ProcessDescription></wps:ProcessDescriptions>
```

A.1.2 PointInPolygonAggregation Operation

```
<wps:ProcessDescriptions xmlns:wps="http://www.opengeospatial.net/wps"><wps:ProcessDescription processVersion="1" storeSupported="false"
statusSupported="false" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink">
        <ows:Identifier>org.n52.wps.server.algorithm.PointInPolygonAggregationAlgorithm</ows:Identifier>
        <ows:Title>Join (Point-In-Polygon) and Aggregation of Spatial Data</ows:Title>
        <ows:Abstract>of GML data containing points with GML data containing polygons based on the spatial"inside" followed by the aggregation of
attribute values.aggregate functions: count, sum, max and min.is possible to keep attributes of the polygon data.
</ows:Abstract>
        <ows:Metadata xlink:title="join"/>
        <ows:Metadata xlink:title="aggregate"/>
        <wps:DataInputs>
            <wps:Input>
                <ows:Identifier>PointFeatures</ows:Identifier>
                <ows:Title>PointFeatures as GML data or reference</ows:Title>
                <ows:Abstract>PointFeatures as GML data or reference</ows:Abstract>
                <wps:ComplexData/>
                <wps:MinimumOccurs>1</wps:MinimumOccurs>
            </wps:Input>
            <wps:Input>
                <ows:Identifier>PolygonFeatures</ows:Identifier>
                <ows:Title>PolygonFeatures as GML data or reference</ows:Title>
                <ows:Abstract>PolygonFeatures</ows:Abstract>
                <wps:ComplexData/>
                <wps:MinimumOccurs>1</wps:MinimumOccurs>
            </wps:Input>
            <wps:Input>
                <ows:Identifier>Keep</ows:Identifier>
                <ows:Title>Attributes of the polygon data to keep</ows:Title>
                <ows:Abstract>Attributes of the polygon data to keep, comma separated (string can be empty)</ows:Abstract>
                <wps:LiteralData>
                    <ows:DataType ows:reference="xs:string"/>
                    <ows:AllowedValues>
                        <ows:Value/>
                    </ows:AllowedValues>
                </wps:LiteralData>
                <wps:MinimumOccurs>1</wps:MinimumOccurs>
            </wps:Input>
            <wps:Input>
                <ows:Identifier>Count</ows:Identifier>
                <ows:Title>Attributes to count</ows:Title>
                <ows:Abstract>Attributes to count, comma separated (string can be empty)</ows:Abstract>
                <wps:LiteralData>
                    <ows:DataType ows:reference="xs:string"/>
                    <ows:AllowedValues>
                        <ows:Value/>
                    </ows:AllowedValues>
                </wps:LiteralData>
                <wps:MinimumOccurs>1</wps:MinimumOccurs>
            </wps:Input>
            <wps:Input>
                <ows:Identifier>Sum</ows:Identifier>
                <ows:Title>Attributes to sum</ows:Title>
                <ows:Abstract>Attributes to sum, comma separated (string can be empty)</ows:Abstract>
                <wps:LiteralData>
                    <ows:DataType ows:reference="xs:string"/>
                    <ows:AllowedValues>
                        <ows:Value/>
                    </ows:AllowedValues>
                </wps:LiteralData>
                <wps:MinimumOccurs>1</wps:MinimumOccurs>
            </wps:Input>
            <wps:Input>
                <ows:Identifier>Max</ows:Identifier>
                <ows:Title>Attributes to find Max</ows:Title>
                <ows:Abstract>Attributes to find the maximum values, comma separated (string can be empty)
                </ows:Abstract>
                <wps:LiteralData>
                    <ows:DataType ows:reference="xs:string"/>
                    <ows:AllowedValues>
                        <ows:Value/>
                    </ows:AllowedValues>
                </wps:LiteralData>
                <wps:MinimumOccurs>1</wps:MinimumOccurs>
            </wps:Input>
            <wps:Input>
                <ows:Identifier>Min</ows:Identifier>
                <ows:Title>Attributes to find Min</ows:Title>
```

```
            <ows:Abstract>Attributes to find the minimum values, comma separated (string can be empty)
            </ows:Abstract>
            <wps:LiteralData>
               <ows:DataType ows:reference="xs:string"/>
               <ows:AllowedValues>
                  <ows:Value/>
               </ows:AllowedValues>
            </wps:LiteralData>
            <wps:MinimumOccurs>1</wps:MinimumOccurs>
         </wps:Input>
      </wps:DataInputs>
      <wps:ProcessOutputs>
         <wps:Output>
            <ows:Identifier>URL_To_GMLResult</ows:Identifier>
            <ows:Title>URL to the Result file</ows:Title>
            <ows:Abstract>URL result file </ows:Abstract>
            <wps:LiteralOutput>
               <ows:DataType ows:reference="xs:string"/>
               <ows:AllowedValues>
                  <ows:Value/>
               </ows:AllowedValues>
            </wps:LiteralOutput>
         </wps:Output>
      </wps:ProcessOutputs>
   </wps:ProcessDescription></wps:ProcessDescriptions>
```

A.1.3 Normalization Operation

```
<wps:ProcessDescriptions xmlns:wps="http://www.opengeospatial.net/wps"><wps:ProcessDescription processVersion="1" storeSupported="false"
statusSupported="false" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink">
      <ows:Identifier>org.n52.wps.server.algorithm.NormalisationByFixedValueAlgorithm</ows:Identifier>
      <ows:Title>Normalisation of Spatial Data</ows:Title>
      <ows:Abstract>Normalisation of GML data. A set of attribute values ("Attributes") can be normalised by a fixed value ("NormaliseBy"). It is not
allowed to normalise by 0.</ows:Abstract>
      <ows:Metadata xlink:title="normalise"/>
      <wps:DataInputs>
         <wps:Input>
            <ows:Identifier>InputFeatures</ows:Identifier>
            <ows:Title>InputFeatures</ows:Title>
            <ows:Abstract>URL of the InputFeatures</ows:Abstract>
            <wps:ComplexValueReference/>
            <wps:MinimumOccurs>1</wps:MinimumOccurs>
         </wps:Input>
         <wps:Input>
            <ows:Identifier>NormaliseBy</ows:Identifier>
            <ows:Title>Value to normalise by</ows:Title>
            <ows:Abstract>Fixed Value to normalise by</ows:Abstract>
            <wps:LiteralData>
               <ows:DataType ows:reference="xs:double"/>
               <ows:AllowedValues>
                  <ows:Value/>
               </ows:AllowedValues>
            </wps:LiteralData>
            <wps:MinimumOccurs>1</wps:MinimumOccurs>
         </wps:Input>
         <wps:Input>
            <ows:Identifier>Attributes</ows:Identifier>
            <ows:Title>Attributes to normalise</ows:Title>
            <ows:Abstract>Attributes to normalise, comma separated</ows:Abstract>
            <wps:LiteralData>
               <ows:DataType ows:reference="xs:string"/>
               <ows:AllowedValues>
                  <ows:Value/>
               </ows:AllowedValues>
            </wps:LiteralData>
            <wps:MinimumOccurs>1</wps:MinimumOccurs>
         </wps:Input>
      </wps:DataInputs>
      <wps:ProcessOutputs>
         <wps:Output>
            <ows:Identifier>URL_To_GMLResult</ows:Identifier>
            <ows:Title>URL to the Result file</ows:Title>
            <ows:Abstract>URL result file </ows:Abstract>
            <wps:LiteralOutput>
               <ows:DataType ows:reference="xs:string"/>
               <ows:AllowedValues>
                  <ows:Value/>
               </ows:AllowedValues>
```

```
            </wps:LiteralOutput>
        </wps:Output>
    </wps:ProcessOutputs>
</wps:ProcessDescription></wps:ProcessDescriptions>
```

A.1.4 Deviations

In D 4.2.2 the algorithm for the Join operation has been presented as a ‘Left Outer Join’. For the reasons given in

A.2 PS Classification Service

A.2.1 Classify Operation

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessDescriptions xmlns="http://www.opengeospatial.net/wps" xmlns:wps="http://www.opengeospatial.net/wps"
xmlns:ows="http://www.opengeospatial.net/ows" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengeospatial.net/wps http://naturegis.jrc.it:8080/schemas/wps/0.4.0/wpsAll.xsd">
    <ProcessDescription processVersion="1" storeSupported="true" statusSupported="false">
        <ows:Identifier>ClassifyAlgorithm</ows:Identifier>
        <ows:Title>Classification of a Feature Collection</ows:Title>
        <ows:Abstract>This operation calculates the class ranges for an input Feature Collection (in GML format); classification is performed on a specific
attribute specified as input parameter. Only numeric attributes are supported. Other mandatory input parameters are the classifcation method to be applied
to calculate ranges, and the number or the step of classes. Four different output options are available. When the "store" parameter is set to "true",  an URL
is returned that refers an accessible resource on the server; such resource can be an SLD or an XML document that defines the classes. When the "store"
parameter is set to "false", a SLD or an XML document is returned, that defines the classes.</ows:Abstract>
        <DataInputs>
            <Input>
                <ows:Identifier>FeatureCollection</ows:Identifier>
                <ows:Title>The feature collection to be classified.</ows:Title>
                <ows:Abstract>Location of the feature collection to be classified.</ows:Abstract>
                <ComplexData defaultFormat="text/xml" defaultEncoding="base64" defaultSchema="http://www.opengis.net/gml">
                    <SupportedComplexData>
                        <Format>text/XML</Format>
                        <Encoding>UTF-8</Encoding>
                        <Schema>http://www.opengis.net/gml</Schema>
                    </SupportedComplexData>
                </ComplexData>
                <MinimumOccurs>1</MinimumOccurs>
            </Input>
            <Input>
                <ows:Identifier>CaName</ows:Identifier>
                <ows:Title>The Name of the attribute to be classified.</ows:Title>
                <ows:Abstract>The Name of the attribute to be classified. The attribute type must be numeric.</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="xs:string"/>
                    <ows:AnyValue/>
                </LiteralData>
                <MinimumOccurs>1</MinimumOccurs>
            </Input>
            <Input>
                <ows:Identifier>CaType</ows:Identifier>
                <ows:Title>The Type of the attribute to be classified.</ows:Title>
                <ows:Abstract>The Type of the attribute to be classified. It must be numeric.</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="xs:string"/>
                    <ows:AllowedValues>
                        <ows:Value>int</ows:Value>
                        <ows:Value>float</ows:Value>
                        <ows:Value>double</ows:Value>
                    </ows:AllowedValues>
                </LiteralData>
                <MinimumOccurs>1</MinimumOccurs>
            </Input>
            <Input>
                <ows:Identifier>NumberClasses</ows:Identifier>
                <ows:Title>Number of classes</ows:Title>
                <ows:Abstract>This parameter specifies the number of intervals that will be used in the classification. This option and the StepClass option
are mutually exclusive.</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="xs:int"/>
                    <ows:AnyValue/>
                </LiteralData>
                <MinimumOccurs>0</MinimumOccurs>
            </Input>
            <Input>
                <!-- currently not allowed-->
```

```
            <ows:Identifier>StepClass</ows:Identifier>
            <ows:Title>The step of each class.</ows:Title>
            <ows:Abstract>This parameter specifies the size of the intervals that will be used in the classification. Available only when the chosen
Classification Method is "equalIntervalequalInterval". This option and the NumberClasses option are mutually exclusive.</ows:Abstract>
            <LiteralData>
               <ows:DataType ows:reference="xs:double"/>
               <ows:AnyValue/>
            </LiteralData>
            <MinimumOccurs>0</MinimumOccurs>
         </Input>
         <Input>
            <ows:Identifier>ClassifyMethod</ows:Identifier>
            <ows:Title>Classification method to be applied.</ows:Title>
            <ows:Abstract>Supported methods are: NaturalBreaks, StandardDeviation, Quantile and EqualInterval.</ows:Abstract>
            <LiteralData>
               <ows:DataType ows:reference="xs:string"/>
               <ows:AllowedValues>
                  <ows:Value>NaturalBreaks</ows:Value>
                  <ows:Value>StandardDeviation</ows:Value>
                  <ows:Value>Quantile</ows:Value>
                  <ows:Value>EqualInterval</ows:Value>
               </ows:AllowedValues>
            </LiteralData>
            <MinimumOccurs>1</MinimumOccurs>
         </Input>
         <Input>
            <ows:Identifier>UpperBoundary</ows:Identifier>
            <ows:Title>The upper boundary of the restricted range of values.</ows:Title>
            <ows:Abstract>This parameter is used to exclude from computation a range of attribute values. All features exceeding the upper boundary
will not be classified. The boundary is not inclusive.</ows:Abstract>
            <LiteralData>
               <ows:DataType ows:reference="xs:double"/>
               <ows:AnyValue/>
            </LiteralData>
            <MinimumOccurs>0</MinimumOccurs>
         </Input>
         <Input>
            <ows:Identifier>LowerBoundary</ows:Identifier>
            <ows:Title>The lower boundary of the restricted range of values.</ows:Title>
            <ows:Abstract>This parameter is used to exclude from computation a range of attribute values. Only features exceeding the lower boundary
will be classified. The boundary is not inclusive.</ows:Abstract>
            <LiteralData>
               <ows:DataType ows:reference="xs:double"/>
               <ows:AnyValue/>
            </LiteralData>
            <MinimumOccurs>0</MinimumOccurs>
         </Input>
         <Input>
            <ows:Identifier>StartColor</ows:Identifier>
            <ows:Title>Start Color</ows:Title>
            <ows:Abstract>This parameter is used for creating the SLD result. It specifies the start color for the color ramp used in the rendering of
classified data. The color encoding must be in hexadecimal format. This parameter is mandatory when results are requested as SLD
document.</ows:Abstract>
            <LiteralData>
               <ows:DataType ows:reference="xs:string"/>
               <ows:AnyValue/>
            </LiteralData>
            <MinimumOccurs>0</MinimumOccurs>
         </Input>
         <Input>
            <ows:Identifier>EndColor</ows:Identifier>
            <ows:Title>End Color</ows:Title>
            <ows:Abstract>This parameter is used for creating the SLD result. It specifies the end color for the color ramp used in the rendering of
classified data. The color encoding must be in hexadecimal format. This parameter is mandatory when results are requested as SLD
document.</ows:Abstract>
            <LiteralData>
               <ows:DataType ows:reference="xs:string"/>
               <ows:AnyValue/>
            </LiteralData>
            <MinimumOccurs>0</MinimumOccurs>
         </Input>
         <Input>
            <ows:Identifier>RenderingServer</ows:Identifier>
            <ows:Title>Select the rendering server.</ows:Title>
            <ows:Abstract>Specifies which kind of rendering server the user prefers.</ows:Abstract>
            <LiteralData>
               <ows:DataType ows:reference="xs:string"/>
               <ows:AllowedValues>
                  <ows:Value>MAS</ows:Value>
```

```
                    <ows:Value>WMS</ows:Value>
                    <ows:Value>Google Earth</ows:Value>
                </ows:AllowedValues>
            </LiteralData>
            <MinimumOccurs>0</MinimumOccurs>
        </Input>
        <Input>
            <ows:Identifier>EPSG</ows:Identifier>
            <ows:Title>The EPSG code associated to the feature collection.</ows:Title>
            <LiteralData>
                <ows:DataType ows:reference="xs:int"/>
                <ows:AnyValue/>
            </LiteralData>
            <MinimumOccurs>0</MinimumOccurs>
        </Input>
    </DataInputs>
    <ProcessOutputs>
        <Output>
            <ows:Identifier>SLD</ows:Identifier>
            <ows:Title>Styled Layer Descriptor</ows:Title>
            <ows:Abstract>The styled layer descriptor resulting from the classification. It can be directly sent to a WMS or MAS server for rendering the
result of the classify operation. Only one kind of output can be specified in a classify request.</ows:Abstract>
            <ComplexOutput defaultFormat="text/XML" defaultEncoding="base64" defaultSchema="http://www.opengis.net/sld">
                <SupportedComplexData>
                    <Format>text/XML</Format>
                    <Encoding>UTF-8</Encoding>
                    <Schema>http://www.opengis.net/sld</Schema>
                </SupportedComplexData>
            </ComplexOutput>
        </Output>
        <Output>
            <ows:Identifier>KML</ows:Identifier>
            <ows:Title>Keyhole Markup Language</ows:Title>
            <ows:Abstract>The KML file resulting from the classification. KML is a file format used to display geographic data in Earth browser such as
Google(TM) Earth, Google(TM) Maps and Google(TM) Maps for Mobile.</ows:Abstract>
            <ComplexOutput defaultFormat="text/XML">
                <SupportedComplexData>
                    <Format>text/XML</Format>
                    <Encoding>UTF-8</Encoding>
                </SupportedComplexData>
            </ComplexOutput>
        </Output>
        <Output>
            <ows:Identifier>Ranges</ows:Identifier>
            <ows:Title>The boundaries of the intervals calculated by the classifiy operation.</ows:Title>
            <ows:Abstract>An XML document containing the intervals of the classification. Only one kind of output can be specified in a classify
request.</ows:Abstract>
            <ComplexOutput defaultFormat="text/XML" defaultEncoding="base64" defaultSchema="localSchema">
                <SupportedComplexData>
                    <Format>text/XML</Format>
                    <Encoding>UTF-8</Encoding>
                    <Schema>localSchema</Schema>
                </SupportedComplexData>
            </ComplexOutput>
        </Output>
    </ProcessOutputs>
  </ProcessDescription>
</ProcessDescriptions>
```

A.2.2 Assign Classes Operation

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessDescriptions xmlns="http://www.opengeospatial.net/wps" xmlns:wps="http://www.opengeospatial.net/wps"
xmlns:ows="http://www.opengeospatial.net/ows" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengeospatial.net/wps http://naturegis.jrc.it:8080/schemas/wps/0.4.0/wpsAll.xsd">
  <ProcessDescription processVersion="1" storeSupported="true" statusSupported="false">
    <ows:Identifier>AssignClassesAlgorithm</ows:Identifier>
    <ows:Title>Assignment of an attribute to a feature collection</ows:Title>
        <ows:Abstract>The operation AssignClasses creates and populates a new attribute in a feature collection. Attribute values are assigned
        according to a set of logical conditions specified in the AssignClauses document (a list of mutually exclusive boolean expressions.) Input
        parameters are: the feature collection to be classified, the name of the new attribute and the document containing
    the logical conditions.</ows:Abstract>
    <!--ows:Metadata xlink:title="assign classes"/>
    <ows:Metadata xlink:title="gml"/-->
    <DataInputs>
      <Input>
        <ows:Identifier>FeatureCollection</ows:Identifier>
        <ows:Title>The Feature Collection to be classified</ows:Title>
```

```
            <ows:Abstract>Location and name of the input GML file</ows:Abstract>
            <ComplexData defaultFormat="text/xml" defaultEncoding="base64" defaultSchema="http://www.opengis.net/gml">
              <SupportedComplexData>
                <Format>text/XML</Format>
                <Encoding>UTF-8</Encoding>
                <Schema>http://www.opengis.net/gml</Schema>
              </SupportedComplexData>
            </ComplexData>
            <MinimumOccurs>1</MinimumOccurs>
          </Input>
          <Input>
            <ows:Identifier>AssignClauses</ows:Identifier>
            <ows:Title>Boolean expressions to be verified for attribute values assignement.</ows:Title>
            <ows:Abstract>A set of mutually exclusive boolean expressions in ogc:filter format. Each clause must specify the attribute value to be assigned
if verified.</ows:Abstract>
            <ComplexData defaultFormat="text/xml" defaultEncoding="base64"
defaultSchema="http://naturegis.jrc.it:8080/schemas/ps/AssignClauses.xsd">
              <SupportedComplexData>
                <Format>text/xml</Format>
                <Encoding>UTF-8</Encoding>
                <Schema>http://naturegis.jrc.it:8080/schemas/ps/AssignClauses.xsd</Schema>
              </SupportedComplexData>
            </ComplexData>
            <MinimumOccurs>1</MinimumOccurs>
          </Input>
          <Input>
            <ows:Identifier>NewType</ows:Identifier>
            <ows:Title>The new attribute type.</ows:Title>
            <ows:Abstract>The type of the new attribute that will be created within the Feature Collection.</ows:Abstract>
            <LiteralData>
              <ows:DataType ows:reference="xs:string"/>
              <ows:AllowedValues>
                <ows:Value>int</ows:Value>
                <ows:Value>double</ows:Value>
                <ows:Value>string</ows:Value>
              </ows:AllowedValues>
            </LiteralData>
            <MinimumOccurs>1</MinimumOccurs>
          </Input>
          <Input>
            <ows:Identifier>NewName</ows:Identifier>
            <ows:Title>The new attribute name.</ows:Title>
            <ows:Abstract>The name of the new attribute that will be created within the Feature Collection.</ows:Abstract>
            <LiteralData>
              <ows:DataType ows:reference="xs:string"/>
              <ows:AnyValue/>
            </LiteralData>
            <MinimumOccurs>1</MinimumOccurs>
          </Input>
          <Input>
            <ows:Identifier>Length</ows:Identifier>
            <ows:Title>The maximum length for values of the new attribute, when its type is string.</ows:Title>
            <ows:Abstract>The maximum length for values of the new attribute, when its type is string.</ows:Abstract>
            <LiteralData>
              <ows:DataType ows:reference="xs:int"/>
              <ows:AnyValue/>
            </LiteralData>
            <MinimumOccurs>0</MinimumOccurs>
          </Input>
          <Input>
            <ows:Identifier>EPSG</ows:Identifier>
            <ows:Title>The EPSG code of the Feature Collection.</ows:Title>
            <LiteralData>
              <ows:DataType ows:reference="xs:int"/>
              <ows:AnyValue/>
            </LiteralData>
            <MinimumOccurs>0</MinimumOccurs>
          </Input>
        </DataInputs>
        <ProcessOutputs>
          <Output>
            <ows:Identifier>GML</ows:Identifier>
            <ows:Title>The new Feature Collection with the new attribute.</ows:Title>
            <ows:Abstract>The new feature collection with the newly added attribute.</ows:Abstract>
            <ComplexOutput defaultFormat="text/XML" defaultEncoding="base64" defaultSchema="http://www.opengis.net/gml">
              <SupportedComplexData>
                <Format>text/XML</Format>
                <Encoding>UTF-8</Encoding>
                <Schema>http://www.opengis.net/gml</Schema>
```

```
            </SupportedComplexData>
          </ComplexOutput>
        </Output>
      </ProcessOutputs>
    </ProcessDescription>
</ProcessDescriptions>
```

A.2.3 Classify and Assign Operation

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessDescriptions xmlns="http://www.opengeospatial.net/wps" xmlns:wps="http://www.opengeospatial.net/wps"
xmlns:ows="http://www.opengeospatial.net/ows" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengeospatial.net/wps http://naturegis.jrc.it:8080/schemas/wps/0.4.0/wpsAll.xsd">
    <ProcessDescription processVersion="1" storeSupported="false" statusSupported="false">
        <ows:Identifier>ClassifyAndAssignAlgorithm</ows:Identifier>
        <ows:Title>This operation classifies a feature collection and then add a new attribute to it.</ows:Title>
        <ows:Abstract>This operation classifies the feature collection and adds a new attribute to it. Basing on an ordered list of values, each interval found
by the classification operation is assigned one item in the list of values</ows:Abstract>
        <!--ows:Metadata xlink:title="classify"/>
        <ows:Metadata xlink:title="gml"/-->
        <DataInputs>
            <Input>
                <ows:Identifier>FeatureCollection</ows:Identifier>
                <ows:Title>Feature Collection</ows:Title>
                        <ows:Abstract>Location and name of the input GML file</ows:Abstract>
                        <ComplexData defaultFormat="text/xml" defaultEncoding="base64" defaultSchema="http://www.opengis.net/gml">
                            <SupportedComplexData>
                                <Format>text/XML</Format>
                                <Encoding>UTF-8</Encoding>
                                <Schema>http://www.opengis.net/gml</Schema>
                            </SupportedComplexData>
                        </ComplexData>
                <MinimumOccurs>1</MinimumOccurs>
            </Input>
            <Input>
                <ows:Identifier>CharacterStringList</ows:Identifier>
                <ows:Title>List of allowed values for the new attribute</ows:Title>
                <ows:Abstract>The list of allowed values for the new attribute. The number and the order of items in the list must be the same as the number
and order of classification intervals specified.
                Moreover, the type of the values must be compliant with the type of the new attribute.</ows:Abstract>
                        <ComplexData defaultFormat="text/xml" defaultEncoding="base64"
defaultSchema="http://naturegis.jrc.it:8080/schemas/ps/CharacterStringList.xsd">
                            <SupportedComplexData>
                                <Format>text/xml</Format>
                                <Encoding>UTF-8</Encoding>
                                <Schema>http://naturegis.jrc.it:8080/schemas/ps/CharacterStringList.xsd</Schema>
                            </SupportedComplexData>
                        </ComplexData>
                <MinimumOccurs>1</MinimumOccurs>
            </Input>
            <Input>
                <ows:Identifier>NewName</ows:Identifier>
                <ows:Title>The Name for the new attribute.</ows:Title>
                <ows:Abstract>The name of the attribute that will be added to the feature collection.</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="xs:string"/>
                    <ows:AnyValue/>
                </LiteralData>
                <MinimumOccurs>1</MinimumOccurs>
            </Input>
            <Input>
                <ows:Identifier>NewType</ows:Identifier>
                <ows:Title>The Type of the new attribute.</ows:Title>
                <ows:Abstract>The type of the attribute that will be added to the feature collection.</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="xs:string"/>
                    <ows:AllowedValues>
                        <ows:Value>int</ows:Value>
                        <ows:Value>float</ows:Value>
                        <ows:Value>double</ows:Value>
                        <ows:Value>string</ows:Value>
                    </ows:AllowedValues>
                </LiteralData>
                <MinimumOccurs>1</MinimumOccurs>
            </Input>
            <Input>
                <ows:Identifier>CaName</ows:Identifier>
                <ows:Title>The name of the attribute to be classified.</ows:Title>
```

```
        <ows:Abstract>The Name of the attribute to be classified. The attribute type must be numeric.</ows:Abstract>
        <LiteralData>
          <ows:DataType ows:reference="xs:string"/>
          <ows:AnyValue/>
        </LiteralData>
        <MinimumOccurs>1</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>CaType</ows:Identifier>
        <ows:Title>The Type of the attribute to be classified.</ows:Title>
        <ows:Abstract>The Type of the attribute to be classified. The attribute type must be numeric.</ows:Abstract>
        <LiteralData>
          <ows:DataType ows:reference="xs:string"/>
          <ows:AllowedValues>
            <ows:Value>int</ows:Value>
            <ows:Value>float</ows:Value>
            <ows:Value>double</ows:Value>
          </ows:AllowedValues>
        </LiteralData>
        <MinimumOccurs>1</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>Length</ows:Identifier>
        <ows:Title>The maximum length for values of the new attribute, when its type is string.</ows:Title>
       <ows:Abstract>The maximum length for values of the new attribute, when its type is string.</ows:Abstract>
        <LiteralData>
          <ows:DataType ows:reference="xs:int"/>
          <ows:AnyValue/>
        </LiteralData>
        <MinimumOccurs>0</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>ClassifyMethod</ows:Identifier>
        <ows:Title>Classification method to be applied.</ows:Title>
        <ows:Abstract>Supported methods are: NaturalBreaks, StandardDeviation, Quantile and EqualInterval.</ows:Abstract>
        <LiteralData>
          <ows:DataType ows:reference="xs:string"/>
          <ows:AllowedValues>
            <ows:Value>NaturalBreaks</ows:Value>
            <ows:Value>StandardDeviation</ows:Value>
            <ows:Value>Quantile</ows:Value>
            <ows:Value>EqualInterval</ows:Value>
          </ows:AllowedValues>
        </LiteralData>
        <MinimumOccurs>1</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>NumberClasses</ows:Identifier>
        <ows:Title>Number of classes</ows:Title>
        <ows:Abstract>This parameter specifies the number of intervals that will be used in the classification.
        This option and the StepClass option are mutually exclusive.</ows:Abstract>
        <LiteralData>
          <ows:DataType ows:reference="xs:int"/>
          <ows:AnyValue/>
        </LiteralData>
        <MinimumOccurs>1</MinimumOccurs>
      </Input>
     <Input>
        <ows:Identifier>EPSG</ows:Identifier>
        <ows:Title>The EPSG code associated to the feature collection.</ows:Title>
        <LiteralData>
                <ows:DataType ows:reference="xs:int"/>
          <ows:AnyValue/>
        </LiteralData>
        <MinimumOccurs>0</MinimumOccurs>
      </Input>
    </DataInputs>
    <ProcessOutputs>
      <Output>
        <ows:Identifier>GML</ows:Identifier>
        <ows:Title>The new feature collection with the new  attribute.</ows:Title>
        <ows:Abstract>The new feature collection with the newly added attribute.</ows:Abstract>
        <ComplexOutput defaultFormat="text/XML" defaultEncoding="base64">
          <SupportedComplexData>
            <Format>text/XML</Format>
            <Encoding>UTF-8</Encoding>
            <Schema>http://www.opengis.net/gml</Schema>
          </SupportedComplexData>
        </ComplexOutput>
```

```
        </Output>
      </ProcessOutputs>
    </ProcessDescription>
</ProcessDescriptions>
```

A.2.4 Deviations

In the actual implementation, some elements, which are conceptually a unique class, have been split into several parameters, which are passed directly as input data.

- OT_ColorRamp (→ StartColor, EndColor)
- OT_Attribute (→ CaName, CaType, Length)
- Boundary (→ upperBoundary, lowerBoundary).

This solution bases on the observation that exchanged data are in these cases less complex and structured; passing parameters as “literal values” has thus avoided using complex data in XML validation and performing validation steps.
Further, all operation support the EPSG as input parameter.

A.3 PS Map-Algebra Service

A.3.1 Local Rating

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessDescriptions xmlns="http://www.opengeospatial.net/wps" xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:wps="http://www.opengeospatial.net/wps" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengeospatial.net/wps http://naturegis.jrc.it:8080/schemas/wps/0.4.0/wpsDescribeProcess.xsd">
  <ProcessDescription processVersion="1" statusSupported="false" storeSupported="true">
    <ows:Identifier>LocalRating</ows:Identifier>
    <ows:Title>Local Rating. It reclasses an input layer according to the input parameters Ranges and Values.</ows:Title>
    <ows:Abstract>The operation LocalRating reclasses an input layer according to the input parameters Ranges and Values. The operation produces a
new raster with the new values. The Ranges defines a set of pixel values that have to be replaced with the new values. The list of Ranges and the list of
Values have to be the same cardinality.</ows:Abstract>
    <DataInputs>
      <Input>
        <ows:Identifier>LayerToBeRated</ows:Identifier>
        <ows:Title>The layer to be rated</ows:Title>
        <ows:Abstract>Location and name of the layer to be rated</ows:Abstract>
        <ComplexData defaultFormat="image/geotiff">
          <SupportedComplexData>
            <Format>image/geotiff</Format>
          </SupportedComplexData>
          <SupportedComplexData>
            <Format>image/ARCGRID</Format>
          </SupportedComplexData>
        </ComplexData>
        <MinimumOccurs>1</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>RangeToValueMappings</ows:Identifier>
        <ows:Title>Range To value mappings</ows:Title>
        <ows:Abstract>List of mappings to transform ranges of the input layer into explicit values of the output layer. The complexValue describes n
mappings, each consisting of a value range, which might be described as a closed or open interval, and a new value into which all values within the given
range shall be transformed.</ows:Abstract>
        <ComplexData defaultEncoding="UTF-8" defaultFormat="text/xml"
defaultSchema="http://naturegis.jrc.it:8080/schemas/ps/mapsInputTypes.xsd"/>
        <MinimumOccurs>1</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>InputLayerDescriptor</ows:Identifier>
        <ows:Title>Description of the InputLayer.</ows:Title>
        <ows:Abstract>An inputLayerDescriptor is an optional input that is used to provide the service with additional informati\on on geopgraphic data
layers that can not be described using the input information only. The description of the input layer contains the layer crs, the noDataValue which will be
ignored during the computation, and any other value that shall be igrnored (e.g. in a ratio it might make sense to ignore the 0 to prevent undefined results).
The use of this element is optional. Default values are the standard NoDataValues for each encoding (Unsigned 8 bit = 0, etc.), the CRS EPSG:4326, and
no values to ignore </ows:Abstract>
        <ComplexData defaultEncoding="text/xml" defaultSchema="http://naturegis.jrc.it:8080/schemas/ps/mapsInputTypes.xsd"/>
        <MinimumOccurs>0</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>OutputLayerDescriptor</ows:Identifier>
        <ows:Title>Definition of the Output Layer</ows:Title>
```

```
            <ows:Abstract>The definition describes the standard NoDataValues if different from the standard for each encoding (Unsigned 8 bit = 0, etc.), and the CRS if different from EPSG:4326.</ows:Abstract>
            <ComplexData defaultEncoding="text/xml" defaultSchema="http://naturegis.jrc.it:8080/schemas/ps/mapsInputTypes.xsd"/>
            <MinimumOccurs>0</MinimumOccurs>
          </Input>
        </DataInputs>
        <ProcessOutputs>
          <Output>
            <ows:Identifier>OutLayer</ows:Identifier>
            <ows:Title>The new new layer created by the operation</ows:Title>
            <ows:Abstract>The new layer is built on the information given from the RangeToValueMappings. The outputs are returned either inline this document with base64 encoding, or can be stored as web accessible resource on the server (if store=true). If the result is stored, the Execute response will consist of a XML document that includes the URL for the stored output,
            which the client can use to retrieve those outputs.</ows:Abstract>
            <ComplexOutput defaultEncoding="Unsigned16bit" defaultFormat="image/geotiff">
              <SupportedComplexData>
                <Format>image/geotiff</Format>
                <Encoding>Unsigned8bit</Encoding>
              </SupportedComplexData>
              <SupportedComplexData>
                <Format>image/geotiff</Format>
                <Encoding>Unsigned8bit/Base64</Encoding>
              </SupportedComplexData>
              <SupportedComplexData>
                <Format>image/geotiff</Format>
                <Encoding>Unsigned16bit</Encoding>
              </SupportedComplexData>
              <SupportedComplexData>
                <Format>image/geotiff</Format>
                <Encoding>Unsigned16bit/Base64</Encoding>
              </SupportedComplexData>
              <SupportedComplexData>
                <Format>image/geotiff</Format>
                <Encoding>Signed16bit</Encoding>
              </SupportedComplexData>
              <SupportedComplexData>
                <Format>image/geotiff</Format>
                <Encoding>Signed16bit/Base64</Encoding>
              </SupportedComplexData>
              <SupportedComplexData>
                <Format>image/ARCGRID</Format>
              </SupportedComplexData>
            </ComplexOutput>
          </Output>
          <Output>
            <ows:Identifier>OutLayerBBox</ows:Identifier>
            <ows:Title>The bounding box of the new layer.</ows:Title>
            <ows:Abstract>The bounding box of the new layer resulted from the product.
              This output will be returned as direct response to the request.
            </ows:Abstract>
            <BoundingBoxOutput defaultCRS="EPSG:4326"/>
          </Output>
        </ProcessOutputs>
      </ProcessDescription>
    </ProcessDescriptions>
```

A.3.2 Local Ratio

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessDescriptions xmlns="http://www.opengeospatial.net/wps" xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:wps="http://www.opengeospatial.net/wps" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengeospatial.net/wps http://naturegis.jrc.it:8080/schemas/wps/0.4.0/wpsDescribeProcess.xsd">
  <ProcessDescription processVersion="1" statusSupported="false" storeSupported="true">
    <ows:Identifier>LocalRatio</ows:Identifier>
    <ows:Title>Local Ratio. It performs the ratio of each location's value on a specified input layer and a constant value.</ows:Title>
    <ows:Abstract>The operation LocalRatio computes the ratio of each location's value on a specified input layer and a second input which is a constant value. The inputs of this operation are 1) the dividend and 2) the divisor of a ratio.</ows:Abstract>
    <DataInputs>
      <Input>
        <ows:Identifier>FirstOperatorLayer</ows:Identifier>
        <ows:Title>The layer that is the dividend of the ratio</ows:Title>
        <ows:Abstract>Location and name of the dividend layer.</ows:Abstract>
        <ComplexData defaultFormat="image/geotiff">
          <SupportedComplexData>
            <Format>image/geotiff</Format>
          </SupportedComplexData>
          <SupportedComplexData>
            <Format>image/ARCGRID</Format>
```

```
            </SupportedComplexData>
          </ComplexData>
          <MinimumOccurs>1</MinimumOccurs>
        </Input>
        <Input>
          <ows:Identifier>SecondOperator</ows:Identifier>
          <ows:Title>The divisor number</ows:Title>
          <ows:Abstract>The number by which each value in the DividendLayer shall be divided.</ows:Abstract>
          <LiteralData>
            <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema11-2/#decimal">decimal</ows:DataType>
            <ows:AnyValue/>
          </LiteralData>
          <MinimumOccurs>1</MinimumOccurs>
        </Input>
        <Input>
          <ows:Identifier>InputLayerDescriptor</ows:Identifier>
          <ows:Title>Descriptor of the InputLayer.</ows:Title>
          <ows:Abstract>The description of the input layer contains the layer crs, the noDataValue which will be ignored during the computation, and any
other value that shall be igrnored (e.g. in a ratio it might make sense to ignore the 0 to prevent undefined results). The use of this element is optional.
Default values are the standard NoDataValues for each encoding (Unsigned 8 bit = 0, etc.), the CRS EPSG:4326, and no values to ignore </ows:Abstract>
          <ComplexData defaultEncoding="text/xml" defaultSchema="http://naturegis.jrc.it:8080/schemas/ps/mapsInputTypes.xsd"/>
          <MinimumOccurs>0</MinimumOccurs>
        </Input>
        <Input>
          <ows:Identifier>OutputLayerDefinition</ows:Identifier>
          <ows:Title>Definition of the Output Layer</ows:Title>
          <ows:Abstract>The definition describes the standard NoDataValues if different from the standard for each encoding (Unsigned 8 bit = 0, etc.),
and the CRS if different from EPSG:4326.</ows:Abstract>
          <ComplexData defaultEncoding="text/xml" defaultSchema="http://naturegis.jrc.it:8080/schemas/ps/mapsInputTypes.xsd"/>
          <MinimumOccurs>0</MinimumOccurs>
        </Input>
      </DataInputs>
      <ProcessOutputs>
        <Output>
          <ows:Identifier>OutLayer</ows:Identifier>
          <ows:Title>The new new layer created by the operation</ows:Title>
          <ows:Abstract>The new layer results from the ratio of each cell value of the input layer (dividend) with the number (divisor).
            The outputs are returned either inline this document with base64 encoding, or can be stored as web accessible resource on the server (if
store=true).
            If the result is stored, the Execute response will consist of a XML document that includes the URL for the stored output,
          which the client can use to retrieve those outputs.</ows:Abstract>
          <ComplexOutput defaultEncoding="Unsigned16bit" defaultFormat="image/geotiff">
            <SupportedComplexData>
              <Format>image/geotiff</Format>
              <Encoding>Unsigned8bit</Encoding>
            </SupportedComplexData>
            <SupportedComplexData>
              <Format>image/geotiff</Format>
              <Encoding>Unsigned8bit/Base64</Encoding>
            </SupportedComplexData>
            <SupportedComplexData>
              <Format>image/geotiff</Format>
              <Encoding>Unsigned16bit</Encoding>
            </SupportedComplexData>
            <SupportedComplexData>
              <Format>image/geotiff</Format>
              <Encoding>Unsigned16bit/Base64</Encoding>
            </SupportedComplexData>
            <SupportedComplexData>
              <Format>image/geotiff</Format>
              <Encoding>Signed16bit</Encoding>
            </SupportedComplexData>
            <SupportedComplexData>
              <Format>image/geotiff</Format>
              <Encoding>Signed16bit/Base64</Encoding>
            </SupportedComplexData>
            <SupportedComplexData>
              <Format>image/ARCGRID</Format>
            </SupportedComplexData>
          </ComplexOutput>
        </Output>
        <Output>
          <ows:Identifier>OutLayerBBox</ows:Identifier>
          <ows:Title>The bounding layer of the new layer.</ows:Title>
          <ows:Abstract>The bounding box of the new layer resulted from the product.
            This output will be returned as direct response to the request.
          </ows:Abstract>
          <BoundingBoxOutput defaultCRS="EPSG:4326"/>
        </Output>
```

```
    </ProcessOutputs>
  </ProcessDescription>
</ProcessDescriptions>
```

A.3.3 Zonal Sum

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessDescriptions xmlns="http://www.opengeospatial.net/wps" xmlns:wps="http://www.opengeospatial.net/wps"
xmlns:ows="http://www.opengeospatial.net/ows" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengeospatial.net/wps http://naturegis.jrc.it:8080/schemas/wps/0.4.0/wpsDescribeProcess.xsd">
  <ProcessDescription processVersion="1" storeSupported="true" statusSupported="false">
    <ows:Identifier>ZonalSum</ows:Identifier>
    <ows:Title>Zonal Sum. It performs the sum of all locations of an input layer within each zone of a given zonal layer.</ows:Title>
    <ows:Abstract>The operation ZonalSum computes the sum of all locations of an input layer within each zone of a given zonal layer. The inputs of
this operation are 1) the input layer whose location values shall be summed up and 2) the zonal layer. The operation ZonalSum can be used as an
aggregation function. In this case the intended output is no longer a representation of a continuous phenomenon. For that reason the operation provides the
results in tabular format.</ows:Abstract>
    <DataInputs>
      <Input>
        <ows:Identifier>ValueLayer</ows:Identifier>
        <ows:Title>The input layer.</ows:Title>
        <ows:Abstract>Location and name of the value layer whose location values shall be summed up.</ows:Abstract>
        <ComplexData defaultFormat="image/geotiff" defaultEncoding="base64">
          <SupportedComplexData>
            <Format>image/geotiff</Format>
          </SupportedComplexData>
          <SupportedComplexData>
            <Format>image/AAIGrid</Format>
          </SupportedComplexData>
        </ComplexData>
        <MinimumOccurs>1</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>ZonalLayer</ows:Identifier>
        <ows:Title>The Zonal layer.</ows:Title>
        <ows:Abstract>Location and name of the zonal layer.</ows:Abstract>
        <ComplexData defaultFormat="image/geotiff" defaultEncoding="base64">
          <SupportedComplexData>
            <Format>image/geotiff</Format>
          </SupportedComplexData>
          <SupportedComplexData>
            <Format>image/AAIGrid</Format>
          </SupportedComplexData>
        </ComplexData>
        <MinimumOccurs>1</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>InputLayerDescriptorList</ows:Identifier>
        <ows:Title>List of Descriptors for each InputLayer.</ows:Title>
        <ows:Abstract>An inputLayerDescriptorList contains descriptions for each of the layers used as input into this operation. The description of an
input layer contains the layer crs, the noDataValue which will be ignored during the computation, and any other value that shall be igrnored (e.g. in a ratio
it might make sense to ignore the 0 to prevent undefined results). The use of this element is optional. Default values are the standard NoDataValues for
each encoding (Unsigned 8 bit = 0, etc.), the CRS EPSG:4326, and no values to ignore </ows:Abstract>
        <ComplexData defaultEncoding="text/xml" defaultSchema="http://naturegis.jrc.it:8080/schemas/ps/mapsInputTypes.xsd"/>
        <MinimumOccurs>0</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>OutputLayerDefinition</ows:Identifier>
        <ows:Title>Definition of the Output Layer</ows:Title>
        <ows:Abstract>The definition describes the standard NoDataValues if different from the standard for each encoding (Unsigned 8 bit = 0, etc.),
and the CRS if different from EPSG:4326.</ows:Abstract>
        <ComplexData defaultEncoding="text/xml" defaultSchema="http://naturegis.jrc.it:8080/schemas/ps/mapsInputTypes.xsd"/>
        <MinimumOccurs>0</MinimumOccurs>
      </Input>
    </DataInputs>
    <ProcessOutputs>
      <Output>
        <ows:Identifier>Table</ows:Identifier>
        <ows:Title>The resulting table.</ows:Title>
        <ows:Abstract>The resulting Table.
          The outputs can be returned as a direct response to the request.
          Alternatively, the server can be directed to store the result(s) as web accessible resources.
          If the results are stored, the Execute response will consist of a XML document that includes the URL for the stored output,
        which the client can use to retrieve those outputs.</ows:Abstract>
        <ComplexOutput defaultFormat="text/xml" defaultEncoding="base64"
defaultSchema="http://naturegis.jrc.it:8080/schemas/ps/ZonalSumSchema.xsd">
          <SupportedComplexData>
            <Format>text/XML</Format>
            <Encoding>UTF-8</Encoding>
```

```
            <Schema>http://naturegis.jrc.it:8080/schemas/ps/ZonalSumSchema.xsd</Schema>
          </SupportedComplexData>
        </ComplexOutput>
      </Output>
    </ProcessOutputs>
  </ProcessDescription>
</ProcessDescriptions>
```

A.3.4 Zonal Rating

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessDescriptions xmlns="http://www.opengeospatial.net/wps" xmlns:wps="http://www.opengeospatial.net/wps"
xmlns:ows="http://www.opengeospatial.net/ows" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengeospatial.net/wps http://naturegis.jrc.it:8080/schemas/wps/0.4.0/wpsDescribeProcess.xsd">
  <ProcessDescription processVersion="1" storeSupported="true" statusSupported="false">
    <ows:Identifier>ZonalRating</ows:Identifier>
    <ows:Title>Zonal Rating. It creates new values for those locations that spatially correspond with the cells of a zonal layer.</ows:Title>
    <ows:Abstract>The operation ZonalRating creates new values for those locations that spatially correspond with the cells of a zonal layer. The new
value for each location can be either chosen automatically as the value of the corresponding location in the zonal Layer, or they can be given
explicitly.</ows:Abstract>
    <DataInputs>
      <Input>
        <ows:Identifier>ZonalLayer</ows:Identifier>
        <ows:Title>The Zonal layer.</ows:Title>
        <ows:Abstract>Location and name of the binary zonal layer.</ows:Abstract>
        <ComplexData defaultFormat="image/geotiff" defaultEncoding="base64">
          <SupportedComplexData>
            <Format>image/geotiff</Format>
          </SupportedComplexData>
          <SupportedComplexData>
            <Format>image/AAIGrid</Format>
          </SupportedComplexData>
        </ComplexData>
        <MinimumOccurs>1</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>ValueLayer</ows:Identifier>
        <ows:Title>The value layer.</ows:Title>
        <ows:Abstract>Location and name of the value layer. This raster contains the value to be assigned at the zone of the zonal layer.</ows:Abstract>
        <ComplexData defaultFormat="image/geotiff" defaultEncoding="base64">
          <SupportedComplexData>
            <Format>image/geotiff</Format>
          </SupportedComplexData>
          <SupportedComplexData>
            <Format>image/AAIGrid</Format>
          </SupportedComplexData>
        </ComplexData>
        <MinimumOccurs>1</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>ZonalRatingMapping</ows:Identifier>
        <ows:Title>Zonal rating mapping.</ows:Title>
        <ows:Abstract>Mapping of the reclassification.</ows:Abstract>
        <ComplexData defaultEncoding="text/xml" defaultSchema="http://naturegis.jrc.it:8080/schemas/ps/mapsInputTypes.xsd">
          <SupportedComplexData>
            <Format>text/xml</Format>
            <Encoding>UTF-8</Encoding>
            <Schema>http://naturegis.jrc.it:8080/schemas/ps/mapsInputTypes.xsd</Schema>
          </SupportedComplexData>
        </ComplexData>
        <MinimumOccurs>1</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>InputLayerDescriptorList</ows:Identifier>
        <ows:Title>List of Descriptors for each InputLayer.</ows:Title>
        <ows:Abstract>An inputLayerDescriptorList contains descriptions for each of the layers used as input into this operation. The description of an
input layer contains the layer crs, the noDataValue which will be ignored during the computation, and any other value that shall be igrnored (e.g. in a ratio
it might make sense to ignore the 0 to prevent undefined results). The use of this element is optional. Default values are the standard NoDataValues for
each encoding (Unsigned 8 bit = 0, etc.), the CRS EPSG:4326, and no values to ignore </ows:Abstract>
        <ComplexData defaultEncoding="text/xml" defaultSchema="http://naturegis.jrc.it:8080/schemas/ps/mapsInputTypes.xsd"/>
        <MinimumOccurs>0</MinimumOccurs>
      </Input>
      <Input>
        <ows:Identifier>OutputLayerDefinition</ows:Identifier>
        <ows:Title>Definition of the Output Layer</ows:Title>
        <ows:Abstract>The definition describes the standard NoDataValues if different from the standard for each encoding (Unsigned 8 bit = 0, etc.),
and the CRS if different from EPSG:4326.</ows:Abstract>
        <ComplexData defaultEncoding="text/xml" defaultSchema="http://naturegis.jrc.it:8080/schemas/ps/mapsInputTypes.xsd"/>
```

```
        <MinimumOccurs>0</MinimumOccurs>
      </Input>
    </DataInputs>
    <ProcessOutputs>
      <Output>
        <ows:Identifier>OutLayer</ows:Identifier>
        <ows:Title>The new layer.</ows:Title>
        <ows:Abstract>The new layer resulted from the reclassification.
          The outputs can be returned as a direct response to the request.
          Alternatively, the server can be directed to store the result(s) as web accessible resources.
          If the results are stored, the Execute response will consist of a XML document that includes the URL for the stored output,
        which the client can use to retrieve those outputs.</ows:Abstract>
        <ComplexOutput defaultEncoding="Unsigned16bit" defaultFormat="image/geotiff">
          <SupportedComplexData>
            <Format>image/geotiff</Format>
            <Encoding>Unsigned8bit</Encoding>
          </SupportedComplexData>
          <SupportedComplexData>
            <Format>image/geotiff</Format>
            <Encoding>Unsigned8bit/Base64</Encoding>
          </SupportedComplexData>
          <SupportedComplexData>
            <Format>image/geotiff</Format>
            <Encoding>Unsigned16bit</Encoding>
          </SupportedComplexData>
          <SupportedComplexData>
            <Format>image/geotiff</Format>
            <Encoding>Unsigned16bit/Base64</Encoding>
          </SupportedComplexData>
          <SupportedComplexData>
            <Format>image/geotiff</Format>
            <Encoding>Signed16bit</Encoding>
          </SupportedComplexData>
          <SupportedComplexData>
            <Format>image/geotiff</Format>
            <Encoding>Signed16bit/Base64</Encoding>
          </SupportedComplexData>
          <SupportedComplexData>
            <Format>image/ARCGRID</Format>
          </SupportedComplexData>
        </ComplexOutput>
      </Output>
    </ProcessOutputs>
  </ProcessDescription>
</ProcessDescriptions>
```

A.3.5 Deviations

In the service implementation, the LocalFunction operation has been added, with respect to the conceptual design. Such operation gets as input a layer, an operator name (sum, subtraction, ratio and product) and a numeric value, and applies the operator to the two arguments. It is used to implement the LocalRatio operation, and can at the same time perform other local operations.

Appendix B: WSDL Description of Service Chain Instances

B.1 Forest Fire Risk Assessment Service

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="Peunha1" targetNamespace="http://localhost:8080/axis2/services/Peunha1" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/" xmlns:plnk2="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
xmlns:wps="http://www.opengeospatial.net/wps" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:peunha1_exc="http://eu-
orchestra.org/Peunha1/exceptions/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:ns1="http://eu-
orchestra.org/OA/FeatureAccessService" xmlns:ns3="http://localhost:8080/axis2/services/ps" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ns2="urn:service.orchestra.at" xmlns:peunha1_types="http://eu-orchestra.org/Peunha1/types"
xmlns:peunha1="http://localhost:8080/axis2/services/Peunha1" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:import namespace="http://localhost:8080/axis2/services/ps" location="http://orchestra.h07.jrc.it:8090/axis2-
1.2/services/ProcessingService?wsdl"/>
  <wsdl:import namespace="http://eu-orchestra.org/OA/FeatureAccessService"
location="http://orchestra.h07.jrc.it:8080/axis2/services/FeatureAccessService?wsdl"/>
<wsdl:import namespace="urn:service.orchestra.at" location="http://orchestra.h07.jrc.it:8090/axis2-1.1.0/services/RepositoryService?wsdl"/>

  <wsdl:types>
    <xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" targetNamespace="http://localhost:8080/axis2/services/Peunha1"
xmlns="http://localhost:8080/axis2/services/Peunha1" xmlns:xs="http://www.w3.org/2001/XMLSchema">
            <xs:import namespace="http://eu-orchestra.org/Peunha1/types" schemaLocation="Peunha1_types.xsd"/>
            <xs:import namespace="http://eu-orchestra.org/Peunha1/exceptions/" schemaLocation="Peunha1_exc.xsd"/>
            <xs:import namespace="http://www.opengeospatial.net/wps"
schemaLocation="http://www.ogcnetwork.net/schemas/wps/0.4.0/wpsExecute.xsd"/>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="ForestFireRiskClassesRequest">
    <wsdl:part name="request" element="peunha1_types:FFRiskClassesRequest"/>
  </wsdl:message>
  <wsdl:message name="ForestFireDensityRequest">
    <wsdl:part name="request" element="peunha1_types:FFDensityRequest"/>
  </wsdl:message>
  <wsdl:message name="MissingParameterValue">
    <wsdl:part name="fault" element="peunha1_exc:OA_MissingParameterValue"/>
  </wsdl:message>
  <wsdl:message name="InvalidParameterValue">
    <wsdl:part name="fault" element="peunha1_exc:OA_InvalidParameterValue"/>
  </wsdl:message>
  <wsdl:message name="Response">
    <wsdl:part name="out" element="wps:ExecuteResponse"/>
  </wsdl:message>
  <wsdl:message name="ForestFireFrequencyRequest">
    <wsdl:part name="request" element="peunha1_types:FFFrequencyRequest"/>
  </wsdl:message>
  <wsdl:message name="InternalError">
    <wsdl:part name="fault" element="peunha1_exc:OA_InternalError"/>
  </wsdl:message>
  <wsdl:message name="NoApplicableCode">
    <wsdl:part name="fault" element="peunha1_exc:OA_NoApplicableCode"/>
  </wsdl:message>
  <wsdl:portType name="Peunha1">
    <wsdl:operation name="ForestFireFrequency">
      <wsdl:input name="ForestFireFrequencyRequest" message="peunha1:ForestFireFrequencyRequest"/>
      <wsdl:output name="Response" message="peunha1:Response"/>
      <wsdl:fault name="OA_InternalError" message="peunha1:InternalError"/>
      <wsdl:fault name="OA_MissingParameterValue" message="peunha1:MissingParameterValue"/>
      <wsdl:fault name="OA_InvalidParameterValue" message="peunha1:InvalidParameterValue"/>
      <wsdl:fault name="OA_NoApplicableCode" message="peunha1:NoApplicableCode"/>
    </wsdl:operation>
    <wsdl:operation name="ForestFireDensity">
      <wsdl:input name="ForestFireDensityRequest" message="peunha1:ForestFireDensityRequest"/>
      <wsdl:output name="Response" message="peunha1:Response"/>
      <wsdl:fault name="OA_InternalError" message="peunha1:InternalError"/>
      <wsdl:fault name="OA_MissingParameterValue" message="peunha1:MissingParameterValue"/>
      <wsdl:fault name="OA_InvalidParameterValue" message="peunha1:InvalidParameterValue"/>
      <wsdl:fault name="OA_NoApplicableCode" message="peunha1:NoApplicableCode"/>
    </wsdl:operation>
    <wsdl:operation name="ForestFireRiskClasses">
      <wsdl:input name="ForestFireRiskClassesRequest" message="peunha1:ForestFireRiskClassesRequest"/>
      <wsdl:output name="Response" message="peunha1:Response"/>
      <wsdl:fault name="OA_InternalError" message="peunha1:InternalError"/>
      <wsdl:fault name="OA_MissingParameterValue" message="peunha1:MissingParameterValue"/>
      <wsdl:fault name="OA_InvalidParameterValue" message="peunha1:InvalidParameterValue"/>
      <wsdl:fault name="OA_NoApplicableCode" message="peunha1:NoApplicableCode"/>
    </wsdl:operation>
```

```
 </wsdl:portType>
 <wsdl:binding name="Peunha1Service" type="peunha1:Peunha1">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
  <wsdl:operation name="ForestFireFrequency">
  <soap:operation soapAction="ForestFireFrequency" style="document" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   <wsdl:input name="ForestFireFrequencyRequest">
  <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:input>
   <wsdl:output name="Response">
  <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:output>
   <wsdl:fault name="OA_InternalError">
  <soap:fault name="OA_InternalError" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:fault>
   <wsdl:fault name="OA_MissingParameterValue">
  <soap:fault name="OA_MissingParameterValue" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:fault>
   <wsdl:fault name="OA_InvalidParameterValue">
  <soap:fault name="OA_InvalidParameterValue" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:fault>
   <wsdl:fault name="OA_NoApplicableCode">
  <soap:fault name="OA_NoApplicableCode" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="ForestFireDensity">
  <soap:operation soapAction="ForestFireDensity" style="document" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   <wsdl:input name="ForestFireDensityRequest">
  <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:input>
   <wsdl:output name="Response">
  <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:output>
   <wsdl:fault name="OA_InternalError">
  <soap:fault name="OA_InternalError" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:fault>
   <wsdl:fault name="OA_MissingParameterValue">
  <soap:fault name="OA_MissingParameterValue" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:fault>
   <wsdl:fault name="OA_InvalidParameterValue">
  <soap:fault name="OA_InvalidParameterValue" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:fault>
   <wsdl:fault name="OA_NoApplicableCode">
  <soap:fault name="OA_NoApplicableCode" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="ForestFireRiskClasses">
  <soap:operation soapAction="ForestFireRiskClasses" style="document" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   <wsdl:input name="ForestFireRiskClassesRequest">
  <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:input>
   <wsdl:output name="Response">
  <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:output>
   <wsdl:fault name="OA_InternalError">
  <soap:fault name="OA_InternalError" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:fault>
   <wsdl:fault name="OA_MissingParameterValue">
  <soap:fault name="OA_MissingParameterValue" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:fault>
   <wsdl:fault name="OA_InvalidParameterValue">
  <soap:fault name="OA_InvalidParameterValue" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:fault>
   <wsdl:fault name="OA_NoApplicableCode">
  <soap:fault name="OA_NoApplicableCode" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
   </wsdl:fault>
  </wsdl:operation>
 </wsdl:binding>
 <wsdl:service name="Peunha1Service">
  <wsdl:port name="Peunha1" binding="peunha1:Peunha1Service">
  <soap:address location="http://localhost:8080/active-bpel/services/peunha1" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
  </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

Used schema

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- edited with XMLSpy v2006 sp2 U (http://www.altova.com) by administrator (COMMISSIONE EUROPEA CENTRO DI RICERCA) -->
```

```
<xs:schema xmlns="http://eu-orchestra.org/Peunha1/types" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc" xmlns:wps="http://www.opengeospatial.net/wps"
targetNamespace="http://eu-orchestra.org/Peunha1/types">
    <!-- Input Types necessary for each service invoked by the chain -->
    <xs:complexType name="FASForestFireRequest">
        <xs:sequence>
            <xs:element name="endpoint" type="xs:string" minOccurs="0"/>
            <xs:element name="typeName" type="xs:string" default="orchestra:common_forestfire_sample_2003_4326" minOccurs="0"/>
            <xs:element name="geometryPropName" type="xs:string" minOccurs="0"/>
            <xs:element name="datePropName" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="FASNutsRequest">
        <xs:sequence>
            <xs:element name="endpoint" type="xs:string" minOccurs="0"/>
            <xs:element name="typeName" type="xs:string" default="orchestra:nuts3_4326" minOccurs="0"/>
            <xs:element name="geometryPropName" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="FASQuery">
        <xs:sequence>
            <xs:element name="coordinates" type="xs:string"/>
            <xs:element name="startDate" type="xs:string"/>
            <xs:element name="endDate" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="WPSFrequencyRequest">
        <xs:sequence>
            <xs:element name="endpoint" type="xs:string" minOccurs="0"/>
            <xs:element name="process" type="xs:string" default="org.n52.wps.server.algorithm.PointInPolygonAggregationAlgorithm" minOccurs="0"/>
            <xs:element name="keep" type="xs:string" default="" minOccurs="0"/>
            <xs:element name="count" type="xs:string" default="FIREID" minOccurs="0"/>
            <xs:element name="sum" type="xs:string" default="" minOccurs="0"/>
            <xs:element name="max" type="xs:string" default="" minOccurs="0"/>
            <xs:element name="min" type="xs:string" default="" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="WPSDensityRequest">
        <xs:sequence>
            <xs:element name="endpoint" type="xs:string" minOccurs="0"/>
            <xs:element name="process" type="xs:string" default="org.n52.wps.server.algorithm.NormalisationAlgorithm" minOccurs="0"/>
            <xs:element name="normalizeBy" type="xs:string" default="the_geom" minOccurs="0"/>
            <xs:element name="attributes" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="WPSClassifyRequest">
        <xs:sequence>
            <xs:element name="endpoint" type="xs:string" minOccurs="0"/>
            <xs:element name="process" type="xs:string" default="AssignClassesAlgorithm" minOccurs="0"/>
            <xs:element name="newType" type="xs:string" default="classification" minOccurs="0"/>
            <xs:element name="newName" type="xs:string" default="FireHClass" minOccurs="0"/>
            <xs:element name="newAttLength" type="xs:int" default="6"/>
            <xs:element name="assignClauses" type="xs:string" default="http://naturegis.jrc.it:8080/schemas/config/AssignClauses3.xml"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="WPSClassifyAlgRequest">
        <xs:sequence>
            <xs:element name="endpoint" type="xs:string" minOccurs="0"/>
            <xs:element name="process" type="xs:string" default="ClassifyAlgorithm" minOccurs="0"/>
            <xs:element name="classifymethod" type="xs:string" default="StandardDeviation" minOccurs="0"/>
            <xs:element name="numberclasses" type="xs:int"/>
            <xs:element name="caName" type="xs:string"/>
            <xs:element name="caType" type="xs:string"/>
            <xs:element name="StartColor" type="xs:string" default="008800" minOccurs="0"/>
            <xs:element name="EndColor" type="xs:string" default="FF0000" minOccurs="0"/>
            <xs:element name="classifyOutput" type="xs:string" default="SLD" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <!-- Input Types for the three different operations of the chain -->
    <xs:element name="FFFrequencyRequest" type="FFFrequencyRequest"/>
    <xs:complexType name="FFFrequencyRequest">
        <xs:sequence>
            <xs:element name="FASQuery" type="FASQuery"/>
            <xs:element name="FASForestFireRequest" type="FASForestFireRequest"/>
            <xs:element name="FASNutsRequest" type="FASNutsRequest"/>
            <xs:element name="WPSFrequencyRequest" type="WPSFrequencyRequest"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="FFDensityRequest" type="FFDensityRequest"/>
```

```
    <xs:complexType name="FFDensityRequest">
        <xs:sequence>
            <xs:element name="FASQuery" type="FASQuery"/>
            <xs:element name="FASForestFireRequest" type="FASForestFireRequest"/>
            <xs:element name="FASNutsRequest" type="FASNutsRequest"/>
            <xs:element name="WPSFrequencyRequest" type="WPSFrequencyRequest"/>
            <xs:element name="WPSDensityRequest" type="WPSDensityRequest"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="FFRiskClassesRequest" type="FFRiskClassesRequest"/>
    <xs:complexType name="FFRiskClassesRequest">
        <xs:sequence>
            <xs:element name="FASQuery" type="FASQuery"/>
            <xs:element name="FASForestFireRequest" type="FASForestFireRequest"/>
            <xs:element name="FASNutsRequest" type="FASNutsRequest"/>
            <xs:element name="WPSFrequencyRequest" type="WPSFrequencyRequest"/>
            <xs:element name="WPSDensityRequest" type="WPSDensityRequest"/>
            <xs:element name="WPSClassifyRequest" type="WPSClassifyRequest"/>
        </xs:sequence>
    </xs:complexType>
    <!-- Types used within the chain -->

    <xs:element name="BBOX">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="PropertyName"/>
                <xs:element ref="Box"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="Box">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="coordinates"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="Filter">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="BBOX"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="PropertyName" type="xs:string"/>
    <xs:element name="Query">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="Filter"/>
            </xs:sequence>
            <xs:attribute name="typename" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="coordinates" type="xs:string"/>
    <xs:element name="Query4FAS">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="Query"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <!-- Ouput Types for the three different operations of the chain -->
    <!-- here we could use the execute response for all the 3 op -->
    <!--   <xs:element name="Response" type="xs:string"/> -->
    <xs:element name="Response" type="xs:anyType"/>
</xs:schema>


<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns="http://eu-orchestra.org/Peunha1/exceptions/" xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://eu-
orchestra.org/Peunha1/exceptions/">
    <xs:element name="OA_MissingParameterValue" type="OA_MissingParameterValue"/>
    <xs:complexType name="OA_MissingParameterValue">
        <xs:annotation>
            <xs:documentation>Operation request does not include a value of a mandatory parameter.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="parameterName" type="xs:string">
                <xs:annotation>
```

```
                <xs:documentation>name of the missing parameter</xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<xs:element name="OA_InvalidParameterValue" type="OA_InvalidParameterValue"/>
<xs:complexType name="OA_InvalidParameterValue">
    <xs:annotation>
        <xs:documentation>Operation request contains an invalid parameter value.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="parameterName" type="xs:string">
            <xs:annotation>
                <xs:documentation>name of the parameter that contains an invalid value</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="value" type="xs:string">
            <xs:annotation>
                <xs:documentation>value of the parameter that is considered to be invalid</xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<xs:element name="OA_NoApplicableCode" type="OA_NoApplicableCode"/>
<xs:complexType name="OA_NoApplicableCode">
    <xs:annotation>
        <xs:documentation>No other exceptionCode specified by this service and OSI applies to this exception</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="error" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="OA_InternalError" type="OA_InternalError"/>
<xs:complexType name="OA_InternalError">
    <xs:annotation>
        <xs:documentation>A problem occurred in the runtime environment (e.g. out of memory)</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="location" type="OA_CodeLocation" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>gives context information on source code level about the occurrence of the error (optional)</xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<xs:element name="OA_CodeLocation" type="OA_CodeLocation"/>
<xs:complexType name="OA_CodeLocation">
    <xs:annotation>
        <xs:documentation>provides information on source code level that may be relevant to analyse an exception due to an internal
error</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="lineNumber" type="xs:int">
            <xs:annotation>
                <xs:documentation>specifies the line number in the source file in which the error occurred</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="operationName" type="xs:string">
            <xs:annotation>
                <xs:documentation>specifies the name of the operation in which the error occurred</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="serviceType" type="xs:string">
            <xs:annotation>
                <xs:documentation>specifies the service type to which the internal error corresponds</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="sourceFileName" type="xs:string">
            <xs:annotation>
                <xs:documentation>specifies the name of the source file in which the error occurred</xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>
</xs:schema>
```

B.2 Flood Simulation Service

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="Peunha2" targetNamespace="http://localhost:8080/axis2/services/Peunha2" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:wps="http://www.opengeospatial.net/wps" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:peunha2_exc="http://eu-
orchestra.org/Peunha2/exceptions/" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:peunha2_types="http://eu-orchestra.org/Peunha2/types"
xmlns:peunha2="http://localhost:8080/axis2/services/Peunha2" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" targetNamespace="http://localhost:8080/axis2/services/Peunha2"
xmlns="http://localhost:8080/axis2/services/Peunha2" xmlns:xs="http://www.w3.org/2001/XMLSchema">
            <xs:import namespace="http://eu-orchestra.org/Peunha2/types" schemaLocation="Peunha2_types.xsd"/>
            <xs:import namespace="http://eu-orchestra.org/Peunha2/exceptions/" schemaLocation="Peunha2_exc.xsd"/>
            <xs:import namespace="http://www.opengeospatial.net/wps"
schemaLocation="http://www.ogcnetwork.net/schemas/wps/0.4.0/wpsExecute.xsd"/>
    </xs:schema>
  </wsdl:types>

  <wsdl:message name="FloodSimulationRequest">
    <wsdl:part name="request" element="peunha2_types:FloodSimulationRequest"/>
  </wsdl:message>
  <wsdl:message name="FloodSimulationResponse">
    <wsdl:part name="out" element="wps:ExecuteResponse"/>
  </wsdl:message>

  <wsdl:message name="MissingParameterValue">
    <wsdl:part name="fault" element="peunha2_exc:OA_MissingParameterValue"/>
  </wsdl:message>
  <wsdl:message name="InvalidParameterValue">
    <wsdl:part name="fault" element="peunha2_exc:OA_InvalidParameterValue"/>
  </wsdl:message>
  <wsdl:message name="InternalError">
    <wsdl:part name="fault" element="peunha2_exc:OA_InternalError"/>
  </wsdl:message>
  <wsdl:message name="NoApplicableCode">
    <wsdl:part name="fault" element="peunha2_exc:OA_NoApplicableCode"/>
  </wsdl:message>

  <wsdl:portType name="Peunha2">
    <wsdl:operation name="FloodSimulation">
      <wsdl:input name="FloodSimulationRequest" message="peunha2:FloodSimulationRequest"/>
      <wsdl:output name="FloodSimulationResponse" message="peunha2:FloodSimulationResponse"/>
      <wsdl:fault name="OA_InternalError" message="peunha2:InternalError"/>
      <wsdl:fault name="OA_MissingParameterValue" message="peunha2:MissingParameterValue"/>
      <wsdl:fault name="OA_InvalidParameterValue" message="peunha2:InvalidParameterValue"/>
      <wsdl:fault name="OA_NoApplicableCode" message="peunha2:NoApplicableCode"/>
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="Peunha2Service" type="peunha2:Peunha2">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    <wsdl:operation name="FloodSimulation">
    <soap:operation soapAction="FloodSimulation" style="document" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
      <wsdl:input name="FloodSimulationRequest">
    <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
      </wsdl:input>
      <wsdl:output name="FloodSimulationResponse">
    <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
      </wsdl:output>
      <wsdl:fault name="OA_InternalError">
    <soap:fault name="OA_InternalError" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
      </wsdl:fault>
      <wsdl:fault name="OA_MissingParameterValue">
    <soap:fault name="OA_MissingParameterValue" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
      </wsdl:fault>
      <wsdl:fault name="OA_InvalidParameterValue">
    <soap:fault name="OA_InvalidParameterValue" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
      </wsdl:fault>
      <wsdl:fault name="OA_NoApplicableCode">
    <soap:fault name="OA_NoApplicableCode" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="Peunha2Service">
    <wsdl:port name="Peunha2" binding="peunha2:Peunha1Service">
    <soap:address location="http://localhost:8080/active-bpel/services/peunha2" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:port>
  </wsdl:service>
```

```
</wsdl:definitions>
```

Used schema

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- edited with XMLSpy v2006 sp2 U (http://www.altova.com) by administrator (COMMISSIONE EUROPEA CENTRO DI RICERCA) -->
<xs:schema xmlns="http://eu-orchestra.org/Peunha2/types" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc" xmlns:wps="http://www.opengeospatial.net/wps"
xmlns:mas="http://maps.orchestra.jrc.it" targetNamespace="http://eu-orchestra.org/Peunha2/types">
    <!-- Input Types necessary for each service invoked by the chain -->
    <xs:import namespace="http://maps.orchestra.jrc.it" schemaLocation="http://naturegis.jrc.it:8080/schemas/ps/mapsInputTypes.xsd"/>
    <xs:element name="FloodSimulationRequest" type="FloodSimulationRequest"/>
    <xs:complexType name="FloodSimulationRequest">
        <xs:sequence>
            <xs:element name="River" type="River"/>
            <xs:element name="FloodHeight" type="xs:nonNegativeInteger"/>
            <xs:element name="AreaOfInterest" type="BoundingBoxData"/>
            <xs:element ref="mas:OutputLayerDescriptor" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="River">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Rhine"/>
            <xs:enumeration value="Ruhr"/>
            <xs:enumeration value="Moselle"/>
            <xs:enumeration value="Neckar"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:element name="Response" type="xs:anyType"/>
    <xs:complexType name="BoundingBoxData">
        <xs:annotation>
            <xs:documentation>Boundaries of the hazardous event which causes the damage. For this event the damage shall be assessed. The event shall
have the following encoding: event = 1, no event = 0</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="LowerCorner" type="xs:string"/>
            <xs:element name="UpperCorner" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="crs" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="4326"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
</xs:schema>
```

B.3 Damage Assessment Service

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="DamageAssessmentService" targetNamespace="http://localhost:8080/axis2/services/DamageAssessmentService"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:wps="http://www.opengeospatial.net/wps"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:damageAssessmentService_exc="http://eu-
orchestra.org/DamageAssessmentService/exceptions/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:damageAssessmentService_types="http://eu-orchestra.org/DamageAssessmentService/types"
xmlns:damageAssessment="http://localhost:8080/axis2/services/DamageAssessmentService" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

  <wsdl:types>
    <xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://localhost:8080/axis2/services/DamageAssessmentService"
xmlns="http://localhost:8080/axis2/services/DamageAssessmentService" xmlns:xs="http://www.w3.org/2001/XMLSchema">
            <xs:import namespace="http://eu-orchestra.org/DamageAssessmentService/types" schemaLocation="DamageAssessmentService_types.xsd"/>
            <xs:import namespace="http://eu-orchestra.org/DamageAssessmentService/exceptions/"
schemaLocation="DamageAssessmentService_exc.xsd"/>
            <xs:import namespace="http://www.opengeospatial.net/wps"
schemaLocation="http://www.ogcnetwork.net/schemas/wps/0.4.0/wpsExecute.xsd"/>
    </xs:schema>
  </wsdl:types>

  <wsdl:message name="DamageAssessmentRequest">
    <wsdl:part name="request" element="damageAssessmentService_types:DamageAssessmentRequest"/>
  </wsdl:message>
  <wsdl:message name="DamageAssessmentResponse">
    <wsdl:part name="out" element="wps:ExecuteResponse"/>
```

```
</wsdl:message>

<wsdl:message name="MissingParameterValue">
  <wsdl:part name="fault" element="damageAssessmentService_exc:OA_MissingParameterValue"/>
</wsdl:message>
<wsdl:message name="InvalidParameterValue">
  <wsdl:part name="fault" element="damageAssessmentService_exc:OA_InvalidParameterValue"/>
</wsdl:message>
<wsdl:message name="InternalError">
  <wsdl:part name="fault" element="damageAssessmentService_exc:OA_InternalError"/>
</wsdl:message>
<wsdl:message name="NoApplicableCode">
  <wsdl:part name="fault" element="damageAssessmentService_exc:OA_NoApplicableCode"/>
</wsdl:message>


<wsdl:portType name="DamageAssessmentService">
  <wsdl:operation name="DamageAssessment">
    <wsdl:input name="DamageAssessmentRequest" message="damageAssessment:DamageAssessmentRequest"/>
    <wsdl:output name="DamageAssessmentResponse" message="damageAssessment:DamageAssessmentResponse"/>
    <wsdl:fault name="OA_InternalError" message="damageAssessment:InternalError"/>
    <wsdl:fault name="OA_MissingParameterValue" message="damageAssessment:MissingParameterValue"/>
    <wsdl:fault name="OA_InvalidParameterValue" message="damageAssessment:InvalidParameterValue"/>
    <wsdl:fault name="OA_NoApplicableCode" message="damageAssessment:NoApplicableCode"/>
  </wsdl:operation>
</wsdl:portType>


<wsdl:binding name="DamageAssessmentService" type="damageAssessment:DamageAssessmentService">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
  <wsdl:operation name="DamageAssessment">
  <soap:operation soapAction="DamageAssessment" style="document" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    <wsdl:input name="DamageAssessmentRequest">
  <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:input>
    <wsdl:output name="DamageAssessmentResponse">
  <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:output>
    <wsdl:fault name="OA_InternalError">
  <soap:fault name="OA_InternalError" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:fault>
    <wsdl:fault name="OA_MissingParameterValue">
  <soap:fault name="OA_MissingParameterValue" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:fault>
    <wsdl:fault name="OA_InvalidParameterValue">
  <soap:fault name="OA_InvalidParameterValue" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:fault>
    <wsdl:fault name="OA_NoApplicableCode">
  <soap:fault name="OA_NoApplicableCode" use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:fault>
  </wsdl:operation>
</wsdl:binding>

<wsdl:service name="DamageAssessmentService">
  <wsdl:port name="DamageAssessment" binding="damageAssessment:DamageAssessmentService">
  <soap:address location="http://localhost:8080/active-bpel/services/DamageAssessment" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
  </wsdl:port>
</wsdl:service>

</wsdl:definitions>
```

Used schema

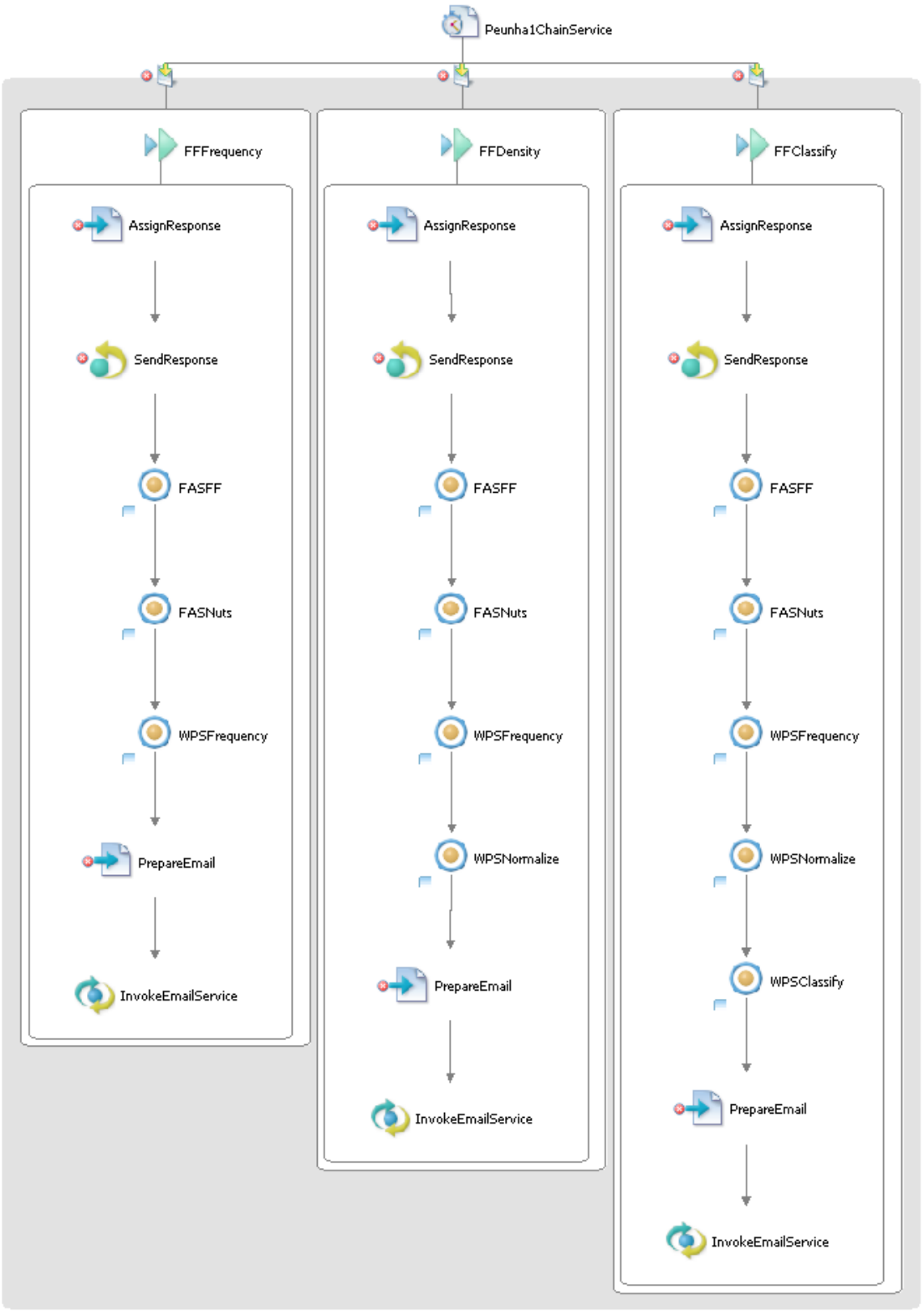
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2006 sp2 U (http://www.altova.com) by administrator (COMMISSIONE EUROPEA CENTRO DI RICERCA) -->
<xs:schema xmlns="http://eu-orchestra.org/DamageAssessmentService/types" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:wps="http://www.opengeospatial.net/wps" targetNamespace="http://eu-orchestra.org/DamageAssessmentService/types">
    <!-- Input Types necessary for each service invoked by the chain -->
    <xs:element name="DamageAssessmentRequest" type="DamageAssessmentRequest"/>
    <xs:complexType name="DamageAssessmentRequest">
        <xs:sequence>
            <xs:element name="Damage" type="DamageType"/>
            <xs:element name="AggregationLevel" type="AggregationLevel"/>
            <xs:element name="BBOXEventExtent" type="BoundingBoxData"/>
            <xs:element name="EventLayer" type="EventLayer"/>
        </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="DamageType">
```

```
        <xs:restriction base="xs:string">
            <xs:enumeration value="NumberOfPeopleAffected"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="AggregationLevel">
        <xs:restriction base="xs:string">
            <xs:enumeration value="NUTS3"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="BoundingBoxData">
        <xs:annotation>
            <xs:documentation>Boundaries of the hazardous event which causes the damage. For this event the damage shall be assessed. The event shall
have the following encoding: event = 1, no event = 0</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="LowerCorner" type="xs:string"/>
            <xs:element name="UpperCorner" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="crs" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="4326"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="EventLayer">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="format" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="image/geotiff"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:schema>
```

Appendix C: BPEL Description of Service Chain Instances

C.1 Forest Fire Risk Assessment Service

Graphical view and BPEL code follows.



```
<?xml version="1.0" encoding="UTF-8"?>
<!--Process Definitionusing ActiveBPEL(tm) Designer Version 4.0.0 (http://www.active-endpoints.com)
-->
<bpel:process xmlns:abx="http://www.activebpel.org/bpel/extension" xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
xmlns:ext="http://www.activebpel.org/2006/09/bpel/extension/query_handling" xmlns:fas="http://eu-orchestra.org/OA/FeatureAccessService"
xmlns:fasTypes="http://eu-orchestra.org/OA/FeatureAccessService/types" xmlns:ns="http://eu-orchestra.org/Peunha1/exceptions/"
xmlns:ns1="urn:service.orchestra.at" xmlns:ns2="http://bpms.intalio.com/tools/webservices/email" xmlns:ns3="http://localhost:8080/axis2/services/ps"
xmlns:ns4="Invalid Document" xmlns:oa="http://eu-orchestra.org/OA/OABasicService/types/1.0" xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:peunha1="http://localhost:8080/axis2/services/Peunha1" xmlns:pt="http://eu-orchestra.org/Peunha1/types"
xmlns:wps="http://www.opengeospatial.net/wps" xmlns:xsd="http://www.w3.org/2001/XMLSchema" ext:createTargetXPath="yes"
name="Peunha1AsynchChain" suppressJoinFailure="yes" targetNamespace="http://test">
  <bpel:extensions>
    <bpel:extension mustUnderstand="yes" namespace="http://www.activebpel.org/2006/09/bpel/extension/query_handling"/>
  </bpel:extensions>
  <bpel:import importType="http://schemas.xmlsoap.org/wsdl/" location="http://naturegis.jrc.it/axis2/services/FeatureAccessService?wsdl"
namespace="http://eu-orchestra.org/OA/FeatureAccessService"/>
  <bpel:import importType="http://schemas.xmlsoap.org/wsdl/" location="http://naturegis.jrc.it/axis2-1.2/services/ProcessingService?wsdl"
namespace="http://localhost:8080/axis2/services/ps"/>
  <bpel:import importType="http://schemas.xmlsoap.org/wsdl/" location="http://naturegis.jrc.it/axis2/services/RepositoryService?wsdl"
namespace="urn:service.orchestra.at"/>
  <bpel:import importType="http://schemas.xmlsoap.org/wsdl/" location="WSDL/APeunha1.wsdl"
namespace="http://localhost:8080/axis2/services/Peunha1"/>
  <bpel:import importType="http://schemas.xmlsoap.org/wsdl/" location="WSDL/EmailWS.wsdl"
namespace="http://bpms.intalio.com/tools/webservices/email"/>
  <bpel:partnerLinks>
    <bpel:partnerLink myRole="Peunha1" name="Peunha1PL" partnerLinkType="peunha1:Peunha1PLT"/>
    <bpel:partnerLink name="WPSPL" partnerLinkType="peunha1:WPSPLT" partnerRole="WPS"/>
    <bpel:partnerLink name="FASPL" partnerLinkType="peunha1:FASPLT" partnerRole="FAS"/>
    <bpel:partnerLink name="RepPL" partnerLinkType="peunha1:RepPLT" partnerRole="Rep"/>
    <bpel:partnerLink name="MSPL" partnerLinkType="peunha1:MSPLT" partnerRole="MS"/>
  </bpel:partnerLinks>
  <bpel:variables>
    <bpel:variable messageType="peunha1:ForestFireFrequencyRequest" name="ForestFireFrequencyRequest"/>
    <bpel:variable messageType="peunha1:Response" name="Response"/>
    <bpel:variable messageType="ns3:executeRequest" name="executeRequestFrequency"/>
    <bpel:variable messageType="ns3:executeResponse" name="executeResponseFrequency"/>
    <bpel:variable messageType="fas:getFeaturesRequest" name="getFFFeaturesRequest"/>
    <bpel:variable messageType="fas:getFeaturesResponse" name="getFFFeaturesResponse"/>
    <bpel:variable messageType="peunha1:ForestFireDensityRequest" name="ForestFireDensityRequest"/>
    <bpel:variable messageType="peunha1:ForestFireRiskClassesRequest" name="ForestFireRiskClassesRequest"/>
    <bpel:variable messageType="ns3:executeRequest" name="executeRequestNormalization"/>
    <bpel:variable messageType="ns3:executeResponse" name="executeResponseNormalization"/>
    <bpel:variable messageType="ns3:executeRequest" name="executeRequestClassification"/>
    <bpel:variable messageType="ns3:executeResponse" name="executeResponseClassification"/>
    <bpel:variable messageType="ns1:inStream" name="FFFeatureCollection"/>
    <bpel:variable messageType="ns1:outUrlAddr" name="FFFeatureColletionUrlAddr"/>
    <bpel:variable messageType="fas:getFeaturesRequest" name="getNutsFeaturesRequest"/>
    <bpel:variable messageType="fas:getFeaturesResponse" name="getNutsFeaturesResponse"/>
    <bpel:variable messageType="ns1:inStream" name="NutsFeatureCollection"/>
    <bpel:variable messageType="ns1:outUrlAddr" name="NutsFeatureColletionUrlAddr"/>
    <bpel:variable name="StatInfoLog" type="xsd:string"/>
    <bpel:variable messageType="ns2:SendEmailRequest" name="SendEmailRequest"/>
    <bpel:variable messageType="ns2:SendEmailResponse" name="SendEmailResponse"/>
  </bpel:variables>
  <bpel:pick createInstance="yes" name="Peunha1ChainService">
    <bpel:onMessage operation="ForestFireFrequency" partnerLink="Peunha1PL" portType="peunha1:Peunha1"
variable="ForestFireFrequencyRequest">
      <bpel:sequence name="FFFrequency">
        <bpel:assign name="AssignResponse">
          <bpel:copy>
            <bpel:from>string(abx:getProcessId())</bpel:from>
            <bpel:to part="out" variable="Response">
              <bpel:query>TaskID</bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:reply name="SendResponse" operation="ForestFireFrequency" partnerLink="Peunha1PL" portType="peunha1:Peunha1"
variable="Response"/>
        <bpel:scope name="FASFF">
          <bpel:sequence>
            <bpel:assign name="PrepareFASQuery">
              <bpel:copy>
                <bpel:from>
                  <bpel:literal>wfs/xml</bpel:literal>
                </bpel:from>
                <bpel:to part="part1" variable="getFFFeaturesRequest">
                  <bpel:query>oa:language</bpel:query>
                </bpel:to>
              </bpel:copy>
```

```
            <bpel:copy>
              <bpel:from>concat('&lt;wfs:Query typeName="', $ForestFireFrequencyRequest.request/FASForestFireRequest/typeName,'"&gt;',
'&lt;ogc:Filter&gt;','&lt;ogc:And&gt;','&lt;ogc:BBOX&gt;','&lt;ogc:PropertyName&gt;',
$ForestFireFrequencyRequest.request/FASForestFireRequest/geometryPropName,'&lt;/ogc:PropertyName&gt;','&lt;gml:Box&gt;','&lt;gml:coordinates&
gt;',
$ForestFireFrequencyRequest.request/FASQuery/coordinates,'&lt;/gml:coordinates&gt;','&lt;/gml:Box&gt;','&lt;/ogc:BBOX&gt;','&lt;ogc:PropertyIsBet
ween&gt;','&lt;ogc:PropertyName&gt;',
$ForestFireFrequencyRequest.request/FASForestFireRequest/datePropName,'&lt;/ogc:PropertyName&gt;','&lt;ogc:LowerBoundary&gt;&lt;ogc:Literal&
gt;', $ForestFireFrequencyRequest.request/FASQuery/startDate
,'&lt;/ogc:Literal&gt;&lt;/ogc:LowerBoundary&gt;','&lt;ogc:UpperBoundary&gt;&lt;ogc:Literal&gt;',
$ForestFireFrequencyRequest.request/FASQuery/endDate,'&lt;/ogc:Literal&gt;&lt;/ogc:UpperBoundary&gt;','&lt;/ogc:PropertyIsBetween&gt;','&lt;/ogc:
And&gt;','&lt;/ogc:Filter&gt;','&lt;/wfs:Query&gt;')</bpel:from>
              <bpel:to part="part1" variable="getFFFeaturesRequest">
                <bpel:query>oa:query</bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:assign name="StatInfoLogging">
            <bpel:copy>
              <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;
"Forest Fire FAS getfeature started the " + year + "-" + month + "-" + day + " at " + hours + ":" + minutes + ":" + seconds;</bpel:from>
              <bpel:to variable="StatInfoLog"/>
            </bpel:copy>
          </bpel:assign>
          <bpel:invoke inputVariable="getFFFeaturesRequest" name="InvokeFAS" operation="getFeatures" outputVariable="getFFFeaturesResponse"
partnerLink="FASPL" portType="fas:FeatureAccessServicePortType"/>
          <bpel:assign name="StatInfoLogging">
            <bpel:copy>
              <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;.getVariableData('StatInfoLog') + "\nForest Fire FAS getfeature completed the " + year + "-" + month + "-" + day + " at " + hours + ":" + minutes +
":" + seconds;</bpel:from>
              <bpel:to variable="StatInfoLog"/>
            </bpel:copy>
          </bpel:assign>
          <bpel:assign name="PrepareFC">
            <bpel:copy>
              <bpel:from part="part1" variable="getFFFeaturesResponse">
                <bpel:query>fasTypes:features</bpel:query>
              </bpel:from>
              <bpel:to part="part1" variable="FFFeatureCollection">
                <bpel:query>in0</bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:invoke inputVariable="FFFeatureCollection" name="InvokeStore" operation="store" outputVariable="FFFeatureColletionUrlAddr"
partnerLink="RepPL" portType="ns1:RepositoryServicePortType"/>
          <bpel:assign name="StatInfoLogging">
            <bpel:copy>
              <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;.getVariableData('StatInfoLog') + "\nForest Fire FC stored the " + year + "-" + month + "-" + day + " at " + hours + ":" + minutes + ":" +
seconds;</bpel:from>
              <bpel:to variable="StatInfoLog"/>
            </bpel:copy>
          </bpel:assign>
        </bpel:sequence>
      </bpel:scope>
      <bpel:scope name="FASNuts">
        <bpel:sequence>
          <bpel:assign name="PrepareFASQuery">
            <bpel:copy>
              <bpel:from>
                <bpel:literal>wfs/xml</bpel:literal>
              </bpel:from>
              <bpel:to part="part1" variable="getNutsFeaturesRequest">
                <bpel:query>oa:language</bpel:query>
              </bpel:to>
```

```
            </bpel:copy>
            <bpel:copy>
              <bpel:from>concat('&lt;wfs:Query typeName="', $ForestFireFrequencyRequest.request/FASNutsRequest/typeName ,'"&gt;',
'&lt;ogc:Filter&gt;','&lt;ogc:BBOX&gt;','&lt;ogc:PropertyName&gt;', $ForestFireFrequencyRequest.request/FASNutsRequest/geometryPropName
,'&lt;/ogc:PropertyName&gt;','&lt;gml:Box&gt;','&lt;gml:coordinates&gt;',
$ForestFireFrequencyRequest.request/FASQuery/coordinates,'&lt;/gml:coordinates&gt;','&lt;/gml:Box&gt;','&lt;/ogc:BBOX&gt;','&lt;/ogc:Filter&gt;','&l
t;/wfs:Query&gt;')</bpel:from>
              <bpel:to part="part1" variable="getNutsFeaturesRequest">
                <bpel:query>oa:query</bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:assign name="StatInfoLogging">
            <bpel:copy>
              <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;.getVariableData('StatInfoLog') + "\nNuts FAS getfeature started the" + year + "-" + month + "-" + day + " at " + hours + ":" + minutes + ":" +
seconds;</bpel:from>
              <bpel:to variable="StatInfoLog"/>
            </bpel:copy>
          </bpel:assign>
          <bpel:invoke inputVariable="getNutsFeaturesRequest" name="InvokeFAS" operation="getFeatures"
outputVariable="getNutsFeaturesResponse" partnerLink="FASPL" portType="fas:FeatureAccessServicePortType"/>
          <bpel:assign name="StatInfoLogging">
            <bpel:copy>
              <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;.getVariableData('StatInfoLog') + "\nNuts FAS getfeature completed the" + year + "-" + month + "-" + day + " at " + hours + ":" + minutes + ":" +
seconds;</bpel:from>
              <bpel:to variable="StatInfoLog"/>
            </bpel:copy>
          </bpel:assign>
          <bpel:assign name="PrepareFC">
            <bpel:copy>
              <bpel:from part="part1" variable="getNutsFeaturesResponse">
                <bpel:query>fasTypes:features</bpel:query>
              </bpel:from>
              <bpel:to part="part1" variable="NutsFeatureCollection">
                <bpel:query>in0</bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:invoke inputVariable="NutsFeatureCollection" name="InvokeStore" operation="store" outputVariable="NutsFeatureColletionUrlAddr"
partnerLink="RepPL" portType="ns1:RepositoryServicePortType"/>
          <bpel:assign name="StatInfoLogging">
            <bpel:copy>
              <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;.getVariableData('StatInfoLog') + "\nNuts FC stored the" + year + "-" + month + "-" + day + " at " + hours + ":" + minutes + ":" +
seconds;</bpel:from>
              <bpel:to variable="StatInfoLog"/>
            </bpel:copy>
          </bpel:assign>
        </bpel:sequence>
      </bpel:scope>
      <bpel:scope name="WPSFrequency">
        <bpel:sequence>
          <bpel:assign name="PrepareWPSRequest">
            <bpel:copy>
              <bpel:from>
                <bpel:literal>WPS</bpel:literal>
              </bpel:from>
              <bpel:to part="request" variable="executeRequestFrequency">
                <bpel:query>@service</bpel:query>
              </bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from>
                <bpel:literal>false</bpel:literal>
              </bpel:from>
```

```
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>@status</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>false</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>@store</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from part="request" variable="ForestFireFrequencyRequest">
        <bpel:query>WPSFrequencyRequest/process</bpel:query>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>ows:Identifier</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>0.4.0</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>@version</bpel:query>
      </bpel:to>
    </bpel:copy>
  </bpel:assign>
  <bpel:assign name="PrepareWPSInput">
    <bpel:copy>
      <bpel:from>
        <bpel:literal>PointFeatures</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>wps:DataInputs/wps:Input[1]/ows:Identifier</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>Point</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>wps:DataInputs/wps:Input[1]/ows:Title</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal></bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from part="part1" variable="FFFeatureColletionUrlAddr">
        <bpel:query>out0</bpel:query>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference/@ows:reference</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>PolygonFeatures</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>wps:DataInputs/wps:Input[2]/ows:Identifier</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>Polygon</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>wps:DataInputs/wps:Input[2]/ows:Title</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
```

```
    <bpel:from>
      <bpel:literal></bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValueReference</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="part1" variable="NutsFeatureColletionUrlAddr">
      <bpel:query>out0</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValueReference/@ows:reference</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Keep</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[3]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Keep</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[3]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="request" variable="ForestFireFrequencyRequest">
      <bpel:query>WPSFrequencyRequest/keep</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>xs:string</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue/@dataType</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Count</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[4]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Count</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[4]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="request" variable="ForestFireFrequencyRequest">
      <bpel:query>WPSFrequencyRequest/count</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[4]/wps:LiteralValue</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>xs:string</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[4]/wps:LiteralValue/@dataType</bpel:query>
    </bpel:to>
  </bpel:copy>
```

```
<bpel:copy>
  <bpel:from>
    <bpel:literal>Sum</bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeRequestFrequency">
    <bpel:query>wps:DataInputs/wps:Input[5]/ows:Identifier</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal>Sum</bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeRequestFrequency">
    <bpel:query>wps:DataInputs/wps:Input[5]/ows:Title</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from part="request" variable="ForestFireFrequencyRequest">
    <bpel:query>WPSFrequencyRequest/sum</bpel:query>
  </bpel:from>
  <bpel:to part="request" variable="executeRequestFrequency">
    <bpel:query>wps:DataInputs/wps:Input[5]/wps:LiteralValue</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal>xs:string</bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeRequestFrequency">
    <bpel:query>wps:DataInputs/wps:Input[5]/wps:LiteralValue/@dataType</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal>Max</bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeRequestFrequency">
    <bpel:query>wps:DataInputs/wps:Input[6]/ows:Identifier</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal>Max</bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeRequestFrequency">
    <bpel:query>wps:DataInputs/wps:Input[6]/ows:Title</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from part="request" variable="ForestFireFrequencyRequest">
    <bpel:query>WPSFrequencyRequest/max</bpel:query>
  </bpel:from>
  <bpel:to part="request" variable="executeRequestFrequency">
    <bpel:query>wps:DataInputs/wps:Input[6]/wps:LiteralValue</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal>xs:string</bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeRequestFrequency">
    <bpel:query>wps:DataInputs/wps:Input[6]/wps:LiteralValue/@dataType</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal>Min</bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeRequestFrequency">
    <bpel:query>wps:DataInputs/wps:Input[7]/ows:Identifier</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal>Min</bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeRequestFrequency">
    <bpel:query>wps:DataInputs/wps:Input[7]/ows:Title</bpel:query>
  </bpel:to>
```

```
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="request" variable="ForestFireFrequencyRequest">
                <bpel:query>WPSFrequencyRequest/min</bpel:query>
              </bpel:from>
              <bpel:to part="request" variable="executeRequestFrequency">
                <bpel:query>wps:DataInputs/wps:Input[7]/wps:LiteralValue</bpel:query>
              </bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from>
                <bpel:literal>xs:string</bpel:literal>
              </bpel:from>
              <bpel:to part="request" variable="executeRequestFrequency">
                <bpel:query>wps:DataInputs/wps:Input[7]/wps:LiteralValue/@dataType</bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:assign name="PrepareWPSOutput">
            <bpel:copy>
              <bpel:from>
                <bpel:literal>URL_to_GML_frequency_Result</bpel:literal>
              </bpel:from>
              <bpel:to part="request" variable="executeRequestFrequency">
                <bpel:query>wps:OutputDefinitions/wps:Output/ows:Identifier</bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:invoke inputVariable="executeRequestFrequency" name="InvokeWPSexecute" operation="execute"
outputVariable="executeResponseFrequency" partnerLink="WPSPL" portType="ns3:ProcessingService"/>
          <bpel:assign name="StatInfoLogging">
            <bpel:copy>
              <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;.getVariableData('StatInfoLog') + "\nAggregation process completed the" + year + "-" + month + "-" + day + " at " + hours + ":" + minutes + ":" +
seconds;</bpel:from>
              <bpel:to variable="StatInfoLog"/>
            </bpel:copy>
          </bpel:assign>
        </bpel:sequence>
      </bpel:scope>
      <bpel:assign name="PrepareEmail">
        <bpel:copy>
          <bpel:from>
            <bpel:literal>intecsorchestra@intecs.it</bpel:literal>
          </bpel:from>
          <bpel:to part="part1" variable="SendEmailRequest">
            <bpel:query>ns2:From</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from part="request" variable="ForestFireFrequencyRequest">
            <bpel:query>emailAddr</bpel:query>
          </bpel:from>
          <bpel:to part="part1" variable="SendEmailRequest">
            <bpel:query>ns2:To</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>concat('Asynchronous Peunha1: chain execution report for task ', string( abx:getProcessId() ) )</bpel:from>
          <bpel:to part="part1" variable="SendEmailRequest">
            <bpel:query>ns2:Subject</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>
              <p>
<anyElement>This is .. </anyElement>
</p>
            </bpel:literal>
          </bpel:from>
          <bpel:to part="part1" variable="SendEmailRequest">
            <bpel:query>ns2:Body</bpel:query>
          </bpel:to>
        </bpel:copy>
```

```
          <bpel:copy>
            <bpel:from>concat('This is to inform you that the submitted task ', string( abx:getProcessId() ) , ' has been completed. The result is available at
', $ForestFireFrequencyRequest.request/NotificationWebPage, '?ID=', string( abx:getProcessId() ), '&amp;amp;Result=',
$executeResponseFrequency.return/wps:ProcessOutputs/wps:Output/wps:LiteralValue)</bpel:from>
            <bpel:to part="part1" variable="SendEmailRequest">
              <bpel:query>ns2:Body/anyElement</bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke inputVariable="SendEmailRequest" name="InvokeEmailService" operation="SendEmail" outputVariable="SendEmailResponse"
partnerLink="MSPL" portType="ns2:EmailWSPortType"/>
      </bpel:sequence>
    </bpel:onMessage>
    <bpel:onMessage operation="ForestFireDensity" partnerLink="Peunha1PL" portType="peunha1:Peunha1" variable="ForestFireDensityRequest">
      <bpel:sequence name="FFDensity">
        <bpel:assign name="AssignResponse">
          <bpel:copy>
            <bpel:from>string(abx:getProcessId())</bpel:from>
            <bpel:to part="out" variable="Response">
              <bpel:query>TaskID</bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:reply name="SendResponse" operation="ForestFireDensity" partnerLink="Peunha1PL" portType="peunha1:Peunha1"
variable="Response"/>
        <bpel:scope name="FASFF">
          <bpel:sequence>
            <bpel:assign name="PrepareFASQuery">
              <bpel:copy>
                <bpel:from>
                  <bpel:literal>wfs/xml</bpel:literal>
                </bpel:from>
                <bpel:to part="part1" variable="getFFFeaturesRequest">
                  <bpel:query>oa:language</bpel:query>
                </bpel:to>
              </bpel:copy>
              <bpel:copy>
                <bpel:from>concat('&lt;wfs:Query typeName="', $ForestFireDensityRequest.request/FASForestFireRequest/typeName,'"&gt;',
'&lt;ogc:Filter&gt;','&lt;ogc:And&gt;','&lt;ogc:BBOX&gt;','&lt;ogc:PropertyName&gt;',
$ForestFireDensityRequest.request/FASForestFireRequest/geometryPropName,'&lt;/ogc:PropertyName&gt;','&lt;gml:Box&gt;','&lt;gml:coordinates&gt;'
,
$ForestFireDensityRequest.request/FASQuery/coordinates,'&lt;/gml:coordinates&gt;','&lt;/gml:Box&gt;','&lt;/ogc:BBOX&gt;','&lt;ogc:PropertyIsBetwee
n&gt;','&lt;ogc:PropertyName&gt;',
$ForestFireDensityRequest.request/FASForestFireRequest/datePropName,'&lt;/ogc:PropertyName&gt;','&lt;ogc:LowerBoundary&gt;&lt;ogc:Literal&gt;'
, $ForestFireDensityRequest.request/FASQuery/startDate
,'&lt;/ogc:Literal&gt;&lt;/ogc:LowerBoundary&gt;','&lt;ogc:UpperBoundary&gt;&lt;ogc:Literal&gt;',
$ForestFireDensityRequest.request/FASQuery/endDate,'&lt;/ogc:Literal&gt;&lt;/ogc:UpperBoundary&gt;','&lt;/ogc:PropertyIsBetween&gt;','&lt;/ogc:An
d&gt;','&lt;/ogc:Filter&gt;','&lt;/wfs:Query&gt;')</bpel:from>
                <bpel:to part="part1" variable="getFFFeaturesRequest">
                  <bpel:query>oa:query</bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
            <bpel:assign name="StatInfoLogging">
              <bpel:copy>
                <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;
"Forest Fire FAS getfeature started the " + year + "-" + month + "-" + day + " at " + hours + ":" + minutes + ":" + seconds;</bpel:from>
                <bpel:to variable="StatInfoLog"/>
              </bpel:copy>
            </bpel:assign>
            <bpel:invoke inputVariable="getFFFeaturesRequest" name="InvokeFAS" operation="getFeatures" outputVariable="getFFFeaturesResponse"
partnerLink="FASPL" portType="fas:FeatureAccessServicePortType"/>
            <bpel:assign name="StatInfoLogging">
              <bpel:copy>
                <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;.getVariableData('StatInfoLog') + "\nForest Fire FAS getfeature completed the " + year + "-" + month + "-" + day + " at " + hours + ":" + minutes +
":" + seconds;</bpel:from>
                <bpel:to variable="StatInfoLog"/>
              </bpel:copy>
            </bpel:assign>
```

```
            <bpel:assign name="PrepareFC">
              <bpel:copy>
                <bpel:from part="part1" variable="getFFFeaturesResponse">
                  <bpel:query>fasTypes:features</bpel:query>
                </bpel:from>
                <bpel:to part="part1" variable="FFFeatureCollection">
                  <bpel:query>in0</bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
            <bpel:invoke inputVariable="FFFeatureCollection" name="InvokeStore" operation="store" outputVariable="FFFeatureColletionUrlAddr"
partnerLink="RepPL" portType="ns1:RepositoryServicePortType"/>
            <bpel:assign name="StatInfoLogging">
              <bpel:copy>
                <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;.getVariableData('StatInfoLog') + "\nForest Fire FC stored the " + year + "-" + month + "-" + day + " at " + hours + ":" + minutes + ":" +
seconds;</bpel:from>
                <bpel:to variable="StatInfoLog"/>
              </bpel:copy>
            </bpel:assign>
          </bpel:sequence>
        </bpel:scope>
        <bpel:scope name="FASNuts">
          <bpel:sequence>
            <bpel:assign name="PrepareFASQuery">
              <bpel:copy>
                <bpel:from>
                  <bpel:literal>wfs/xml</bpel:literal>
                </bpel:from>
                <bpel:to part="part1" variable="getNutsFeaturesRequest">
                  <bpel:query>oa:language</bpel:query>
                </bpel:to>
              </bpel:copy>
              <bpel:copy>
                <bpel:from>concat('&lt;wfs:Query typeName="', $ForestFireDensityRequest.request/FASNutsRequest/typeName ,'"&gt;',
'&lt;ogc:Filter&gt;','&lt;ogc:BBOX&gt;','&lt;ogc:PropertyName&gt;', $ForestFireDensityRequest.request/FASNutsRequest/geometryPropName
,'&lt;/ogc:PropertyName&gt;','&lt;gml:Box&gt;','&lt;gml:coordinates&gt;',
$ForestFireDensityRequest.request/FASQuery/coordinates,'&lt;/gml:coordinates&gt;','&lt;/gml:Box&gt;','&lt;/ogc:BBOX&gt;','&lt;/ogc:Filter&gt;','&lt;/
wfs:Query&gt;')</bpel:from>
                <bpel:to part="part1" variable="getNutsFeaturesRequest">
                  <bpel:query>oa:query</bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
            <bpel:assign name="StatInfoLogging">
              <bpel:copy>
                <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;.getVariableData('StatInfoLog') + "\nNuts FAS getfeature started the" + year + "-" + month + "-" + day + " at " + hours + ":" + minutes + ":" +
seconds;</bpel:from>
                <bpel:to variable="StatInfoLog"/>
              </bpel:copy>
            </bpel:assign>
            <bpel:invoke inputVariable="getNutsFeaturesRequest" name="InvokeFAS" operation="getFeatures"
outputVariable="getNutsFeaturesResponse" partnerLink="FASPL" portType="fas:FeatureAccessServicePortType"/>
            <bpel:assign name="StatInfoLogging">
              <bpel:copy>
                <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;.getVariableData('StatInfoLog') + "\nNuts FAS getfeature completed the" + year + "-" + month + "-" + day + " at " + hours + ":" + minutes + ":" +
seconds;</bpel:from>
                <bpel:to variable="StatInfoLog"/>
              </bpel:copy>
            </bpel:assign>
            <bpel:assign name="PrepareFC">
              <bpel:copy>
                <bpel:from part="part1" variable="getNutsFeaturesResponse">
                  <bpel:query>fasTypes:features</bpel:query>
                </bpel:from>
```

```
                <bpel:to part="part1" variable="NutsFeatureCollection">
                  <bpel:query>in0</bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
            <bpel:invoke inputVariable="NutsFeatureCollection" name="InvokeStore" operation="store" outputVariable="NutsFeatureColletionUrlAddr"
partnerLink="RepPL" portType="ns1:RepositoryServicePortType"/>
            <bpel:assign name="StatInfoLogging">
              <bpel:copy>
                <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;.getVariableData('StatInfoLog') + "\nNuts FC stored the" + year + "-" + month + "-" + day + " at " + hours + ":" + minutes + ":" +
seconds;</bpel:from>
                <bpel:to variable="StatInfoLog"/>
              </bpel:copy>
            </bpel:assign>
          </bpel:sequence>
        </bpel:scope>
        <bpel:scope name="WPSFrequency">
          <bpel:sequence>
            <bpel:assign name="PrepareWPSRequest">
              <bpel:copy>
                <bpel:from>
                  <bpel:literal>WPS</bpel:literal>
                </bpel:from>
                <bpel:to part="request" variable="executeRequestFrequency">
                  <bpel:query>@service</bpel:query>
                </bpel:to>
              </bpel:copy>
              <bpel:copy>
                <bpel:from>
                  <bpel:literal>false</bpel:literal>
                </bpel:from>
                <bpel:to part="request" variable="executeRequestFrequency">
                  <bpel:query>@status</bpel:query>
                </bpel:to>
              </bpel:copy>
              <bpel:copy>
                <bpel:from>
                  <bpel:literal>false</bpel:literal>
                </bpel:from>
                <bpel:to part="request" variable="executeRequestFrequency">
                  <bpel:query>@store</bpel:query>
                </bpel:to>
              </bpel:copy>
              <bpel:copy>
                <bpel:from part="request" variable="ForestFireDensityRequest">
                  <bpel:query>WPSFrequencyRequest/process</bpel:query>
                </bpel:from>
                <bpel:to part="request" variable="executeRequestFrequency">
                  <bpel:query>ows:Identifier</bpel:query>
                </bpel:to>
              </bpel:copy>
              <bpel:copy>
                <bpel:from>
                  <bpel:literal>0.4.0</bpel:literal>
                </bpel:from>
                <bpel:to part="request" variable="executeRequestFrequency">
                  <bpel:query>@version</bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
            <bpel:assign name="PrepareWPSInput">
              <bpel:copy>
                <bpel:from>
                  <bpel:literal>PointFeatures</bpel:literal>
                </bpel:from>
                <bpel:to part="request" variable="executeRequestFrequency">
                  <bpel:query>wps:DataInputs/wps:Input[1]/ows:Identifier</bpel:query>
                </bpel:to>
              </bpel:copy>
              <bpel:copy>
                <bpel:from>
                  <bpel:literal>Point</bpel:literal>
                </bpel:from>
                <bpel:to part="request" variable="executeRequestFrequency">
```

```
      <bpel:query>wps:DataInputs/wps:Input[1]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal></bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="part1" variable="FFFeatureColletionUrlAddr">
      <bpel:query>out0</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference/@ows:reference</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>PolygonFeatures</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[2]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Polygon</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[2]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal></bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValueReference</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="part1" variable="NutsFeatureColletionUrlAddr">
      <bpel:query>out0</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValueReference/@ows:reference</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Keep</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[3]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Keep</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[3]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="request" variable="ForestFireDensityRequest">
      <bpel:query>WPSFrequencyRequest/keep</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>xs:string</bpel:literal>
    </bpel:from>
```

```
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue/@dataType</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Count</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[4]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Count</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[4]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="request" variable="ForestFireDensityRequest">
      <bpel:query>WPSFrequencyRequest/count</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[4]/wps:LiteralValue</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>xs:string</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[4]/wps:LiteralValue/@dataType</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Sum</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[5]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Sum</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[5]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="request" variable="ForestFireDensityRequest">
      <bpel:query>WPSFrequencyRequest/sum</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[5]/wps:LiteralValue</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>xs:string</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[5]/wps:LiteralValue/@dataType</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Max</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[6]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Max</bpel:literal>
```

```
            </bpel:from>
            <bpel:to part="request" variable="executeRequestFrequency">
              <bpel:query>wps:DataInputs/wps:Input[6]/ows:Title</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="request" variable="ForestFireDensityRequest">
              <bpel:query>WPSFrequencyRequest/max</bpel:query>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestFrequency">
              <bpel:query>wps:DataInputs/wps:Input[6]/wps:LiteralValue</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>xs:string</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestFrequency">
              <bpel:query>wps:DataInputs/wps:Input[6]/wps:LiteralValue/@dataType</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>Min</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestFrequency">
              <bpel:query>wps:DataInputs/wps:Input[7]/ows:Identifier</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>Min</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestFrequency">
              <bpel:query>wps:DataInputs/wps:Input[7]/ows:Title</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="request" variable="ForestFireDensityRequest">
              <bpel:query>WPSFrequencyRequest/min</bpel:query>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestFrequency">
              <bpel:query>wps:DataInputs/wps:Input[7]/wps:LiteralValue</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>xs:string</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestFrequency">
              <bpel:query>wps:DataInputs/wps:Input[7]/wps:LiteralValue/@dataType</bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:assign name="PrepareWPSOutput">
          <bpel:copy>
            <bpel:from>
              <bpel:literal>URL_to_GML_frequency_Result</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestFrequency">
              <bpel:query>wps:OutputDefinitions/wps:Output/ows:Identifier</bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke inputVariable="executeRequestFrequency" name="InvokeWPSexecute" operation="execute"
outputVariable="executeResponseFrequency" partnerLink="WPSPL" portType="ns3:ProcessingService"/>
        <bpel:assign name="StatInfoLogging">
          <bpel:copy>
            <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;.getVariableData('StatInfoLog') + "\nAggregation process completed the" + year + "-" + month + "-" + day + " at " + hours + ":" + minutes + ":" +
seconds;</bpel:from>
            <bpel:to variable="StatInfoLog"/>
          </bpel:copy>
        </bpel:assign>
      </bpel:sequence>
```

```
</bpel:scope>
<bpel:scope name="WPSNormalize">
  <bpel:sequence>
    <bpel:assign name="PrepareWPSRequest">
      <bpel:copy>
        <bpel:from>
          <bpel:literal>WPS</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeRequestNormalization">
          <bpel:query>@service</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>false</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeRequestNormalization">
          <bpel:query>@status</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>false</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeRequestNormalization">
          <bpel:query>@store</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from part="request" variable="ForestFireDensityRequest">
          <bpel:query>WPSDensityRequest/process</bpel:query>
        </bpel:from>
        <bpel:to part="request" variable="executeRequestNormalization">
          <bpel:query>ows:Identifier</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>0.4.0</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeRequestNormalization">
          <bpel:query>@version</bpel:query>
        </bpel:to>
      </bpel:copy>
    </bpel:assign>
    <bpel:assign name="PrepareWPSInput">
      <bpel:copy>
        <bpel:from>
          <bpel:literal>InputFeatures</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeRequestNormalization">
          <bpel:query>wps:DataInputs/wps:Input[1]/ows:Identifier</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>Features</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeRequestNormalization">
          <bpel:query>wps:DataInputs/wps:Input[1]/ows:Title</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal></bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeRequestNormalization">
          <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from part="return" variable="executeResponseFrequency">
          <bpel:query>wps:ProcessOutputs/wps:Output/wps:LiteralValue</bpel:query>
        </bpel:from>
        <bpel:to part="request" variable="executeRequestNormalization">
          <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference/@ows:reference</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
```

```
            <bpel:from>
              <bpel:literal>NormaliseBy</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestNormalization">
              <bpel:query>wps:DataInputs/wps:Input[2]/ows:Identifier</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>Normalise</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestNormalization">
              <bpel:query>wps:DataInputs/wps:Input[2]/ows:Title</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="request" variable="ForestFireDensityRequest">
              <bpel:query>WPSDensityRequest/normalizeBy</bpel:query>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestNormalization">
              <bpel:query>wps:DataInputs/wps:Input[2]/wps:LiteralValue</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>xs:string</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestNormalization">
              <bpel:query>wps:DataInputs/wps:Input[2]/wps:LiteralValue/@dataType</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>Attributes</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestNormalization">
              <bpel:query>wps:DataInputs/wps:Input[3]/ows:Identifier</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>Attributes</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestNormalization">
              <bpel:query>wps:DataInputs/wps:Input[3]/ows:Title</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="request" variable="ForestFireDensityRequest">
              <bpel:query>WPSDensityRequest/attributes</bpel:query>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestNormalization">
              <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>xs:string</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestNormalization">
              <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue/@dataType</bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:assign name="PrepareWPSOutput">
          <bpel:copy>
            <bpel:from>
              <bpel:literal>URL_to_GML_density_Result</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestNormalization">
              <bpel:query>wps:OutputDefinitions/wps:Output/ows:Identifier</bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke inputVariable="executeRequestNormalization" name="InvokeWPSexecute" operation="execute"
outputVariable="executeResponseNormalization" partnerLink="WPSPL" portType="ns3:ProcessingService"/>
        <bpel:assign name="StatInfoLogging">
          <bpel:copy>
```

```
                <bpel:from expressionLanguage="urn:active-endpoints:expression-language:javascript1.5">d = new Date();&#13;=
d.getUTCFullYear().toString();&#13;= (d.getUTCMonth()+1).toString();&#13;= d.getUTCDate().toString();&#13;= (d.getUTCHours() +
1).toString();&#13;= d.getUTCMinutes().toString();&#13;= d.getUTCSeconds().toString();&#13;
&#13;(month.length == 1) { month = "0" + month };&#13;(day.length == 1) { day = "0" + day };&#13;(hours.length == 1) { hours = "0" + hours
};&#13;(minutes.length == 1) {minutes = "0" + minutes};&#13;(seconds.length == 1) {seconds = "0" + seconds};&#13;
&#13;.getVariableData('StatInfoLog') + "\nNormalization process completed the" + year + "-" + month + "-" + day + " at " + hours + ":" + minutes + ":" +
seconds;</bpel:from>
                <bpel:to variable="StatInfoLog"/>
              </bpel:copy>
            </bpel:assign>
          </bpel:sequence>
        </bpel:scope>
        <bpel:assign name="PrepareEmail">
          <bpel:copy>
            <bpel:from>
              <bpel:literal>intecsorchestra@intecs.it</bpel:literal>
            </bpel:from>
            <bpel:to part="part1" variable="SendEmailRequest">
              <bpel:query>ns2:From</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="request" variable="ForestFireDensityRequest">
              <bpel:query>emailAddr</bpel:query>
            </bpel:from>
            <bpel:to part="part1" variable="SendEmailRequest">
              <bpel:query>ns2:To</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>concat('Asynchronous Peunha1: chain execution report for task ', string( abx:getProcessId() ) )</bpel:from>
            <bpel:to part="part1" variable="SendEmailRequest">
              <bpel:query>ns2:Subject</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>
                <p>
<anyElement>This is .. </anyElement>
</p>
              </bpel:literal>
            </bpel:from>
            <bpel:to part="part1" variable="SendEmailRequest">
              <bpel:query>ns2:Body</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>concat('This is to inform you that the submitted task ', string( abx:getProcessId() ) , ' has been completed. The result is available at
', $ForestFireDensityRequest.request/NotificationWebPage , '?ID=', string( abx:getProcessId() ), '&amp;Result=',
$executeResponseNormalization.return/wps:ProcessOutputs/wps:Output/wps:LiteralValue)</bpel:from>
            <bpel:to part="part1" variable="SendEmailRequest">
              <bpel:query>ns2:Body/anyElement</bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke inputVariable="SendEmailRequest" name="InvokeEmailService" operation="SendEmail" outputVariable="SendEmailResponse"
partnerLink="MSPL" portType="ns2:EmailWSPortType"/>
      </bpel:sequence>
    </bpel:onMessage>
    <bpel:onMessage operation="ForestFireRiskClasses" partnerLink="Peunha1PL" portType="peunha1:Peunha1"
variable="ForestFireRiskClassesRequest">
      <bpel:sequence name="FFClassify">
        <bpel:assign name="AssignResponse">
          <bpel:copy>
            <bpel:from>string(abx:getProcessId())</bpel:from>
            <bpel:to part="out" variable="Response">
              <bpel:query>TaskID</bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:reply name="SendResponse" operation="ForestFireRiskClasses" partnerLink="Peunha1PL" portType="peunha1:Peunha1"
variable="Response"/>
        <bpel:scope name="FASFF">
          <bpel:sequence>
            <bpel:assign name="PrepareFASQuery">
              <bpel:copy>
                <bpel:from>
                  <bpel:literal>wfs/xml</bpel:literal>
```

```
                </bpel:from>
                <bpel:to part="part1" variable="getFFFeaturesRequest">
                  <bpel:query>oa:language</bpel:query>
                </bpel:to>
              </bpel:copy>
              <bpel:copy>
                <bpel:from>concat('&lt;wfs:Query typeName="', $ForestFireRiskClassesRequest.request/FASForestFireRequest/typeName,'"&gt;',
'&lt;ogc:Filter&gt;','&lt;ogc:And&gt;','&lt;ogc:BBOX&gt;','&lt;ogc:PropertyName&gt;',
$ForestFireRiskClassesRequest.request/FASForestFireRequest/geometryPropName,'&lt;/ogc:PropertyName&gt;','&lt;gml:Box&gt;','&lt;gml:coordinates
&gt;',
$ForestFireRiskClassesRequest.request/FASQuery/coordinates,'&lt;/gml:coordinates&gt;','&lt;/gml:Box&gt;','&lt;/ogc:BBOX&gt;','&lt;ogc:PropertyIsBe
tween&gt;','&lt;ogc:PropertyName&gt;',
$ForestFireRiskClassesRequest.request/FASForestFireRequest/datePropName,'&lt;/ogc:PropertyName&gt;','&lt;ogc:LowerBoundary&gt;&lt;ogc:Literal
&gt;', $ForestFireRiskClassesRequest.request/FASQuery/startDate
,'&lt;/ogc:Literal&gt;&lt;/ogc:LowerBoundary&gt;','&lt;ogc:UpperBoundary&gt;&lt;ogc:Literal&gt;',
$ForestFireRiskClassesRequest.request/FASQuery/endDate,'&lt;/ogc:Literal&gt;&lt;/ogc:UpperBoundary&gt;','&lt;/ogc:PropertyIsBetween&gt;','&lt;/og
c:And&gt;','&lt;/ogc:Filter&gt;','&lt;/wfs:Query&gt;')</bpel:from>
                <bpel:to part="part1" variable="getFFFeaturesRequest">
                  <bpel:query>oa:query</bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
            <bpel:invoke inputVariable="getFFFeaturesRequest" name="InvokeFAS" operation="getFeatures" outputVariable="getFFFeaturesResponse"
partnerLink="FASPL" portType="fas:FeatureAccessServicePortType"/>
            <bpel:assign name="PrepareFC">
              <bpel:copy>
                <bpel:from part="part1" variable="getFFFeaturesResponse">
                  <bpel:query>fasTypes:features</bpel:query>
                </bpel:from>
                <bpel:to part="part1" variable="FFFeatureCollection">
                  <bpel:query>in0</bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
            <bpel:invoke inputVariable="FFFeatureCollection" name="InvokeStore" operation="store" outputVariable="FFFeatureColletionUrlAddr"
partnerLink="RepPL" portType="ns1:RepositoryServicePortType"/>
          </bpel:sequence>
        </bpel:scope>
        <bpel:scope name="FASNuts">
          <bpel:sequence>
            <bpel:assign name="PrepareFASQuery">
              <bpel:copy>
                <bpel:from>
                  <bpel:literal>wfs/xml</bpel:literal>
                </bpel:from>
                <bpel:to part="part1" variable="getNutsFeaturesRequest">
                  <bpel:query>oa:language</bpel:query>
                </bpel:to>
              </bpel:copy>
              <bpel:copy>
                <bpel:from>concat('&lt;wfs:Query typeName="', $ForestFireRiskClassesRequest.request/FASNutsRequest/typeName ,'"&gt;',
'&lt;ogc:Filter&gt;','&lt;ogc:BBOX&gt;','&lt;ogc:PropertyName&gt;', $ForestFireRiskClassesRequest.request/FASNutsRequest/geometryPropName
,'&lt;/ogc:PropertyName&gt;','&lt;gml:Box&gt;','&lt;gml:coordinates&gt;',
$ForestFireRiskClassesRequest.request/FASQuery/coordinates,'&lt;/gml:coordinates&gt;','&lt;/gml:Box&gt;','&lt;/ogc:BBOX&gt;','&lt;/ogc:Filter&gt;','
&lt;/wfs:Query&gt;')</bpel:from>
                <bpel:to part="part1" variable="getNutsFeaturesRequest">
                  <bpel:query>oa:query</bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
            <bpel:invoke inputVariable="getNutsFeaturesRequest" name="InvokeFAS" operation="getFeatures"
outputVariable="getNutsFeaturesResponse" partnerLink="FASPL" portType="fas:FeatureAccessServicePortType"/>
            <bpel:assign name="PrepareFC">
              <bpel:copy>
                <bpel:from part="part1" variable="getNutsFeaturesResponse">
                  <bpel:query>fasTypes:features</bpel:query>
                </bpel:from>
                <bpel:to part="part1" variable="NutsFeatureCollection">
                  <bpel:query>in0</bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
            <bpel:invoke inputVariable="NutsFeatureCollection" name="InvokeStore" operation="store" outputVariable="NutsFeatureColletionUrlAddr"
partnerLink="RepPL" portType="ns1:RepositoryServicePortType"/>
          </bpel:sequence>
        </bpel:scope>
        <bpel:scope name="WPSFrequency">
          <bpel:sequence>
            <bpel:assign name="PrepareWPSRequest">
```

```
    <bpel:copy>
      <bpel:from>
        <bpel:literal>WPS</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>@service</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>false</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>@status</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>false</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>@store</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from part="request" variable="ForestFireRiskClassesRequest">
        <bpel:query>WPSFrequencyRequest/process</bpel:query>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>ows:Identifier</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>0.4.0</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>@version</bpel:query>
      </bpel:to>
    </bpel:copy>
  </bpel:assign>
  <bpel:assign name="PrepareWPSInput">
    <bpel:copy>
      <bpel:from>
        <bpel:literal>PointFeatures</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>wps:DataInputs/wps:Input[1]/ows:Identifier</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>Point</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>wps:DataInputs/wps:Input[1]/ows:Title</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal></bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from part="part1" variable="FFFeatureColletionUrlAddr">
        <bpel:query>out0</bpel:query>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
        <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference/@ows:reference</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>PolygonFeatures</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestFrequency">
```

```
      <bpel:query>wps:DataInputs/wps:Input[2]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Polygon</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[2]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal></bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValueReference</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="part1" variable="NutsFeatureColletionUrlAddr">
      <bpel:query>out0</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValueReference/@ows:reference</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Keep</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[3]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Keep</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[3]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="request" variable="ForestFireRiskClassesRequest">
      <bpel:query>WPSFrequencyRequest/keep</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>xs:string</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue/@dataType</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Count</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[4]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Count</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[4]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="request" variable="ForestFireRiskClassesRequest">
      <bpel:query>WPSFrequencyRequest/count</bpel:query>
    </bpel:from>
```

```
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[4]/wps:LiteralValue</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>xs:string</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[4]/wps:LiteralValue/@dataType</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Sum</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[5]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Sum</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[5]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="request" variable="ForestFireRiskClassesRequest">
      <bpel:query>WPSFrequencyRequest/sum</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[5]/wps:LiteralValue</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>xs:string</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[5]/wps:LiteralValue/@dataType</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Max</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[6]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Max</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[6]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="request" variable="ForestFireRiskClassesRequest">
      <bpel:query>WPSFrequencyRequest/max</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[6]/wps:LiteralValue</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>xs:string</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestFrequency">
      <bpel:query>wps:DataInputs/wps:Input[6]/wps:LiteralValue/@dataType</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Min</bpel:literal>
```

```
            </bpel:from>
            <bpel:to part="request" variable="executeRequestFrequency">
              <bpel:query>wps:DataInputs/wps:Input[7]/ows:Identifier</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>Min</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestFrequency">
              <bpel:query>wps:DataInputs/wps:Input[7]/ows:Title</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="request" variable="ForestFireRiskClassesRequest">
              <bpel:query>WPSFrequencyRequest/min</bpel:query>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestFrequency">
              <bpel:query>wps:DataInputs/wps:Input[7]/wps:LiteralValue</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>xs:string</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestFrequency">
              <bpel:query>wps:DataInputs/wps:Input[7]/wps:LiteralValue/@dataType</bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:assign name="PrepareWPSOutput">
          <bpel:copy>
            <bpel:from>
              <bpel:literal>URL_to_GML_frequency_Result</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestFrequency">
              <bpel:query>wps:OutputDefinitions/wps:Output/ows:Identifier</bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke inputVariable="executeRequestFrequency" name="InvokeWPSexecute" operation="execute"
outputVariable="executeResponseFrequency" partnerLink="WPSPL" portType="ns3:ProcessingService"/>
      </bpel:sequence>
    </bpel:scope>
    <bpel:scope name="WPSNormalize">
      <bpel:sequence>
        <bpel:assign name="PrepareWPSRequest">
          <bpel:copy>
            <bpel:from>
              <bpel:literal>WPS</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestNormalization">
              <bpel:query>@service</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>false</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestNormalization">
              <bpel:query>@status</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>false</bpel:literal>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestNormalization">
              <bpel:query>@store</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="request" variable="ForestFireRiskClassesRequest">
              <bpel:query>WPSDensityRequest/process</bpel:query>
            </bpel:from>
            <bpel:to part="request" variable="executeRequestNormalization">
              <bpel:query>ows:Identifier</bpel:query>
            </bpel:to>
          </bpel:copy>
```

```
    <bpel:copy>
      <bpel:from>
        <bpel:literal>0.4.0</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestNormalization">
        <bpel:query>@version</bpel:query>
      </bpel:to>
    </bpel:copy>
  </bpel:assign>
  <bpel:assign name="PrepareWPSInput">
    <bpel:copy>
      <bpel:from>
        <bpel:literal>InputFeatures</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestNormalization">
        <bpel:query>wps:DataInputs/wps:Input[1]/ows:Identifier</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>Features</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestNormalization">
        <bpel:query>wps:DataInputs/wps:Input[1]/ows:Title</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal></bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestNormalization">
        <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from part="return" variable="executeResponseFrequency">
        <bpel:query>wps:ProcessOutputs/wps:Output/wps:LiteralValue</bpel:query>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestNormalization">
        <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference/@ows:reference</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>NormaliseBy</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestNormalization">
        <bpel:query>wps:DataInputs/wps:Input[2]/ows:Identifier</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>Normalise</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestNormalization">
        <bpel:query>wps:DataInputs/wps:Input[2]/ows:Title</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from part="request" variable="ForestFireRiskClassesRequest">
        <bpel:query>WPSDensityRequest/normalizeBy</bpel:query>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestNormalization">
        <bpel:query>wps:DataInputs/wps:Input[2]/wps:LiteralValue</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>xs:string</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestNormalization">
        <bpel:query>wps:DataInputs/wps:Input[2]/wps:LiteralValue/@dataType</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>Attributes</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestNormalization">
```

```
                <bpel:query>wps:DataInputs/wps:Input[3]/ows:Identifier</bpel:query>
              </bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from>
                <bpel:literal>Attributes</bpel:literal>
              </bpel:from>
              <bpel:to part="request" variable="executeRequestNormalization">
                <bpel:query>wps:DataInputs/wps:Input[3]/ows:Title</bpel:query>
              </bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="request" variable="ForestFireRiskClassesRequest">
                <bpel:query>WPSDensityRequest/attributes</bpel:query>
              </bpel:from>
              <bpel:to part="request" variable="executeRequestNormalization">
                <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue</bpel:query>
              </bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from>
                <bpel:literal>xs:string</bpel:literal>
              </bpel:from>
              <bpel:to part="request" variable="executeRequestNormalization">
                <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue/@dataType</bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:assign name="PrepareWPSOutput">
            <bpel:copy>
              <bpel:from>
                <bpel:literal>URL_to_GML_density_Result</bpel:literal>
              </bpel:from>
              <bpel:to part="request" variable="executeRequestNormalization">
                <bpel:query>wps:OutputDefinitions/wps:Output/ows:Identifier</bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:invoke inputVariable="executeRequestNormalization" name="InvokeWPSexecute" operation="execute"
outputVariable="executeResponseNormalization" partnerLink="WPSPL" portType="ns3:ProcessingService"/>
        </bpel:sequence>
      </bpel:scope>
      <bpel:scope name="WPSClassify">
        <bpel:sequence>
          <bpel:assign name="PrepareWPSRequest">
            <bpel:copy>
              <bpel:from>
                <bpel:literal>WPS</bpel:literal>
              </bpel:from>
              <bpel:to part="request" variable="executeRequestClassification">
                <bpel:query>@service</bpel:query>
              </bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from>
                <bpel:literal>false</bpel:literal>
              </bpel:from>
              <bpel:to part="request" variable="executeRequestClassification">
                <bpel:query>@status</bpel:query>
              </bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from>
                <bpel:literal>true</bpel:literal>
              </bpel:from>
              <bpel:to part="request" variable="executeRequestClassification">
                <bpel:query>@store</bpel:query>
              </bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="request" variable="ForestFireRiskClassesRequest">
                <bpel:query>WPSClassifyRequest/process</bpel:query>
              </bpel:from>
              <bpel:to part="request" variable="executeRequestClassification">
                <bpel:query>ows:Identifier</bpel:query>
              </bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from>
```

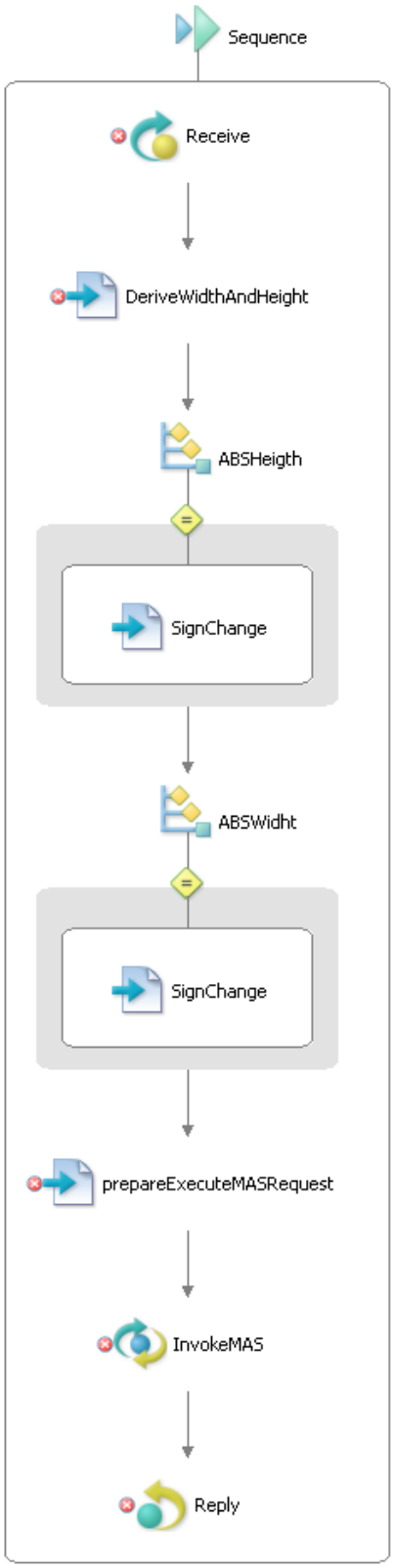
```
        <bpel:literal>0.4.0</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestClassification">
        <bpel:query>@version</bpel:query>
      </bpel:to>
    </bpel:copy>
  </bpel:assign>
  <bpel:assign name="PrepareWPSInput">
    <bpel:copy>
      <bpel:from>
        <bpel:literal>FeatureCollection</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestClassification">
        <bpel:query>wps:DataInputs/wps:Input[1]/ows:Identifier</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>FeatureCollection</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestClassification">
        <bpel:query>wps:DataInputs/wps:Input[1]/ows:Title</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from part="return" variable="executeResponseNormalization">
        <bpel:query>wps:ProcessOutputs/wps:Output/wps:LiteralValue</bpel:query>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestClassification">
        <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference/@ows:reference</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>AssignClauses</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestClassification">
        <bpel:query>wps:DataInputs/wps:Input[2]/ows:Identifier</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>AssignClauses</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestClassification">
        <bpel:query>wps:DataInputs/wps:Input[2]/ows:Title</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from part="request" variable="ForestFireRiskClassesRequest">
        <bpel:query>WPSClassifyRequest/assignClauses</bpel:query>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestClassification">
        <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValueReference/@ows:reference</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>NewType</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestClassification">
        <bpel:query>wps:DataInputs/wps:Input[3]/ows:Identifier</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>NewType</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestClassification">
        <bpel:query>wps:DataInputs/wps:Input[3]/ows:Title</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from part="request" variable="ForestFireRiskClassesRequest">
        <bpel:query>WPSClassifyRequest/newType</bpel:query>
      </bpel:from>
      <bpel:to part="request" variable="executeRequestClassification">
        <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue</bpel:query>
      </bpel:to>
```

```
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>xs:string</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestClassification">
      <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue/@dataType</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>NewName</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestClassification">
      <bpel:query>wps:DataInputs/wps:Input[4]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>NewName</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestClassification">
      <bpel:query>wps:DataInputs/wps:Input[4]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="request" variable="ForestFireRiskClassesRequest">
      <bpel:query>WPSClassifyRequest/newName</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestClassification">
      <bpel:query>wps:DataInputs/wps:Input[4]/wps:LiteralValue</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>xs:string</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestClassification">
      <bpel:query>wps:DataInputs/wps:Input[4]/wps:LiteralValue/@dataType</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Length</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestClassification">
      <bpel:query>wps:DataInputs/wps:Input[5]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>Length</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestClassification">
      <bpel:query>wps:DataInputs/wps:Input[5]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="request" variable="ForestFireRiskClassesRequest">
      <bpel:query>WPSClassifyRequest/newAttLength</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestClassification">
      <bpel:query>wps:DataInputs/wps:Input[5]/wps:LiteralValue</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>xs:int</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeRequestClassification">
      <bpel:query>wps:DataInputs/wps:Input[5]/wps:LiteralValue/@dataType</bpel:query>
    </bpel:to>
  </bpel:copy>
</bpel:assign>
<bpel:assign name="PrepareWPSOutput">
  <bpel:copy>
    <bpel:from>
      <bpel:literal>GML</bpel:literal>
    </bpel:from>
```

```
                <bpel:to part="request" variable="executeRequestClassification">
                  <bpel:query>wps:OutputDefinitions/wps:Output/ows:Identifier</bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
            <bpel:invoke inputVariable="executeRequestClassification" name="InvokeWPSexecute" operation="execute"
outputVariable="executeResponseClassification" partnerLink="WPSPL" portType="ns3:ProcessingService"/>
          </bpel:sequence>
        </bpel:scope>
        <bpel:assign name="PrepareEmail">
          <bpel:copy>
            <bpel:from>
              <bpel:literal>intecsorchestra@intecs.it</bpel:literal>
            </bpel:from>
            <bpel:to part="part1" variable="SendEmailRequest">
              <bpel:query>ns2:From</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="request" variable="ForestFireRiskClassesRequest">
              <bpel:query>emailAddr</bpel:query>
            </bpel:from>
            <bpel:to part="part1" variable="SendEmailRequest">
              <bpel:query>ns2:To</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>concat('Asynchronous Peunha1: chain execution report for task ', string( abx:getProcessId() ) )</bpel:from>
            <bpel:to part="part1" variable="SendEmailRequest">
              <bpel:query>ns2:Subject</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>
              <bpel:literal>
                <p>
<anyElement>This is .. </anyElement>
</p>
              </bpel:literal>
            </bpel:from>
            <bpel:to part="part1" variable="SendEmailRequest">
              <bpel:query>ns2:Body</bpel:query>
            </bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from>concat('This is to inform you that the submitted task ', string( abx:getProcessId() ) , ' has been completed. The result is available at
', $ForestFireRiskClassesRequest.request/NotificationWebPage , '?ID=', string( abx:getProcessId() ), '&amp;amp;Result=',
$executeResponseClassification.return/wps:ProcessOutputs/wps:Output/wps:ComplexValueReference/@ows:reference)</bpel:from>
            <bpel:to part="part1" variable="SendEmailRequest">
              <bpel:query>ns2:Body/anyElement</bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke inputVariable="SendEmailRequest" name="InvokeEmailService" operation="SendEmail" outputVariable="SendEmailResponse"
partnerLink="MSPL" portType="ns2:EmailWSPortType"/>
      </bpel:sequence>
    </bpel:onMessage>
  </bpel:pick>
</bpel:process>
```

C.2 Flood Simulation Service

Graphical view and BPEL code follows.



```
<?xml version="1.0" encoding="UTF-8"?>
<!--Process Definitionusing ActiveBPEL(tm) Designer Version 4.0.0 (http://www.active-endpoints.com)
-->
<bpel:process xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
xmlns:ext="http://www.activebpel.org/2006/09/bpel/extension/query_handling" xmlns:fas="http://eu-orchestra.org/OA/FeatureAccessService"
xmlns:fasTypes="http://eu-orchestra.org/OA/FeatureAccessService/types" xmlns:maps="http://maps.orchestra.jrc.it" xmlns:oab="http://eu-
orchestra.org/OA/OABasicService/types/1.0" xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:peunha2="http://localhost:8080/axis2/services/Peunha2" xmlns:ps="http://localhost:8080/axis2/services/ps" xmlns:pt="http://eu-
orchestra.org/Peunha2/types" xmlns:tifrep="urn:service.orchestra.at" xmlns:wps="http://www.opengeospatial.net/wps"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" ext:createTargetXPath="yes" name="wcs-test" suppressJoinFailure="yes"
targetNamespace="http://wcs-test">
  <bpel:extensions>
    <bpel:extension mustUnderstand="yes" namespace="http://www.activebpel.org/2006/09/bpel/extension/query_handling"/>
  </bpel:extensions>
  <bpel:import importType="http://schemas.xmlsoap.org/wsdl/" location="WSDL/Peunha2.wsdl"
namespace="http://localhost:8080/axis2/services/Peunha2"/>
  <bpel:import importType="http://schemas.xmlsoap.org/wsdl/" location="http://213.173.101.11:8090/axis2/services/NewFeatureAccessService?wsdl"
namespace="http://eu-orchestra.org/OA/FeatureAccessService"/>
  <bpel:import importType="http://schemas.xmlsoap.org/wsdl/" location="http://naturegis.jrc.it/axis2-1.1.0/services/TifRepositoryService?wsdl"
namespace="urn:service.orchestra.at"/>
  <bpel:import importType="http://schemas.xmlsoap.org/wsdl/" location="http://naturegis.jrc.it/axis2-1.2/services/ProcessingService?wsdl"
namespace="http://localhost:8080/axis2/services/ps"/>
  <bpel:import importType="http://www.w3.org/2001/XMLSchema" location="WSDL/masInputTypes.xsd" namespace="http://maps.orchestra.jrc.it"/>
  <bpel:partnerLinks>
    <bpel:partnerLink myRole="FloodSimulator" name="FloodSimulationPL" partnerLinkType="peunha2:FloodSimulationPLT"/>
    <bpel:partnerLink name="WPSPL" partnerLinkType="peunha2:WPSPLT" partnerRole="WPS"/>
  </bpel:partnerLinks>
  <bpel:variables>
    <bpel:variable messageType="peunha2:FloodSimulationRequest" name="FloodSimulationRequest"/>
    <bpel:variable messageType="peunha2:FloodSimulationResponse" name="FloodSimulationResponse"/>
    <bpel:variable name="Width" type="xsd:integer"/>
    <bpel:variable name="Height" type="xsd:integer"/>
    <bpel:variable name="minX" type="xsd:string"/>
    <bpel:variable name="minY" type="xsd:string"/>
    <bpel:variable name="maxX" type="xsd:string"/>
    <bpel:variable name="maxY" type="xsd:string"/>
    <bpel:variable messageType="ps:executeRequest" name="executeMASRequest"/>
    <bpel:variable messageType="ps:executeResponse" name="executeMASResponse"/>
    <bpel:variable name="wcsReference" type="xsd:string"/>
  </bpel:variables>
  <bpel:sequence>
    <bpel:receive createInstance="yes" operation="FloodSimulation" partnerLink="FloodSimulationPL" portType="peunha2:Peunha2"
variable="FloodSimulationRequest"/>
    <bpel:assign name="DeriveWidthAndHeight">
      <bpel:copy>
        <bpel:from>round(&amp;#13;
(&amp;#13;(substring-before ($FloodSimulationRequest.request/AreaOfInterest/UpperCorner, ' ')) -&amp;#13;(substring-before
($FloodSimulationRequest.request/AreaOfInterest/LowerCorner, ' '))&amp;#13;
) &amp;#13;0.00217330619)</bpel:from>
        <bpel:to variable="Width"/>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>round(&amp;#13;
(&amp;#13;(substring-after ($FloodSimulationRequest.request/AreaOfInterest/UpperCorner, ' ')) -&amp;#13;(substring-after
($FloodSimulationRequest.request/AreaOfInterest/LowerCorner, ' '))&amp;#13;
) div 0.00217330619&amp;#13;
)</bpel:from>
        <bpel:to variable="Height"/>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>substring-before( $FloodSimulationRequest.request/AreaOfInterest/LowerCorner ,' ')</bpel:from>
        <bpel:to variable="minX"/>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>substring-before ( $FloodSimulationRequest.request/AreaOfInterest/UpperCorner , ' ')</bpel:from>
        <bpel:to variable="maxX"/>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>substring-after( $FloodSimulationRequest.request/AreaOfInterest/LowerCorner , ' ')</bpel:from>
        <bpel:to variable="minY"/>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>substring-after( $FloodSimulationRequest.request/AreaOfInterest/UpperCorner , ' ')</bpel:from>
        <bpel:to variable="maxY"/>
      </bpel:copy>
    </bpel:assign>
    <bpel:if name="ABSHeigth">
      <bpel:condition>$Height &amp;lt; 0</bpel:condition>
      <bpel:assign name="SignChange">
```

```
          <bpel:copy>
            <bpel:from>- $Height</bpel:from>
            <bpel:to variable="Height"/>
          </bpel:copy>
        </bpel:assign>
      </bpel:if>
      <bpel:if name="ABSWidht">
        <bpel:condition>$Width &lt; 0</bpel:condition>
        <bpel:assign name="SignChange">
          <bpel:copy>
            <bpel:from>- $Width</bpel:from>
            <bpel:to variable="Width"/>
          </bpel:copy>
        </bpel:assign>
      </bpel:if>
      <bpel:assign name="PrepareWCSQuery">
        <bpel:copy>
          <bpel:from>concat (&#13;
'http://naturegis.jrc.it:8080/geoserver/wcs?REQUEST=GetCoverage&amp;amp;Service=WCS&amp;amp;version=1.0.0&amp;amp;Coverage=orchestra:',
$FloodSimulationRequest.request/River , '&amp;amp;format=GeoTIFF&amp;amp;CRS=EPSG:', $FloodSimulationRequest.request/AreaOfInterest/@crs
,'&amp;amp;WIDTH=', $Width, '&amp;amp;HEIGHT=', $Height, '&amp;amp;BBOX=', $minX, ',' , $minY, ',', $maxX, ',' , $maxY )</bpel:from>
          <bpel:to variable="wcsReference"/>
        </bpel:copy>
      </bpel:assign>
      <bpel:assign name="prepareExecuteMASRequest">
        <bpel:copy>
          <bpel:from>
            <bpel:literal>WPS</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>@service</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>false</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>@status</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>true</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>@store</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>LocalRating</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>ows:Identifier</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>0.4.0</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>@version</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>LayerToBeRated</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>wps:DataInputs/wps:Input[1]/ows:Identifier</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal></bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>wps:DataInputs/wps:Input[1]/ows:Title</bpel:query>
```

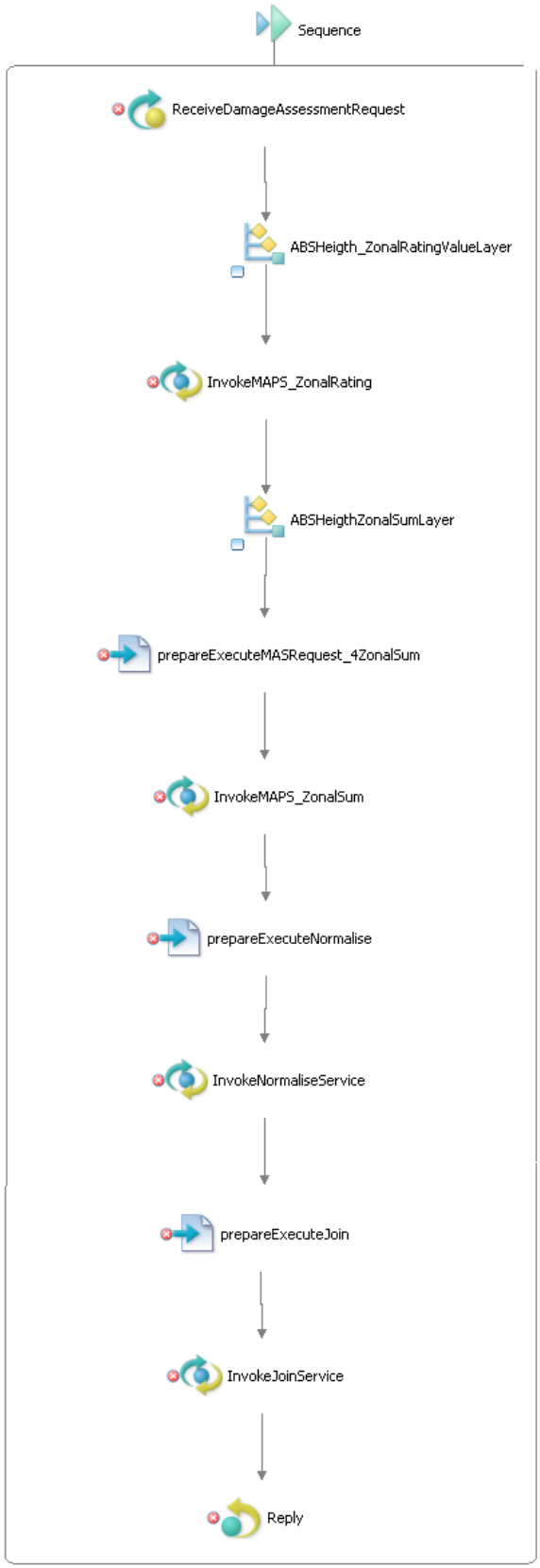
```
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal></bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest">
          <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from variable="wcsReference"/>
        <bpel:to part="request" variable="executeMASRequest">
          <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference/@ows:reference</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>RangeToValueMappings</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest">
          <bpel:query>wps:DataInputs/wps:Input[2]/ows:Identifier</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal></bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest">
          <bpel:query>wps:DataInputs/wps:Input[2]/ows:Title</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>
            <maps:RangeToValueMappings>
    <maps:Mapping>
    <maps:Range rangeClosure="open-closed">
        <maps:MinimumValue/>
        <maps:MaximumValue/>
    </maps:Range>
    <maps:Value>1</maps:Value>
    </maps:Mapping>
    <maps:Mapping>
    <maps:Range rangeClosure="open">
        <maps:MinimumValue/>
        <maps:MaximumValue/>
    </maps:Range>
    <maps:Value>0</maps:Value>
    </maps:Mapping>
</maps:RangeToValueMappings>
          </bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest">
          <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValue/maps:RangeToValueMappings</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from part="request" variable="FloodSimulationRequest">
          <bpel:query>FloodHeight</bpel:query>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest">

<bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValue/maps:RangeToValueMappings/maps:Mapping[1]/maps:Range/maps:MaximumValue</bp
el:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from part="request" variable="FloodSimulationRequest">
          <bpel:query>FloodHeight</bpel:query>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest">

<bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValue/maps:RangeToValueMappings/maps:Mapping[2]/maps:Range/maps:MinimumValue</bp
el:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
```

```
            <bpel:literal>InputLayerDescriptor</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>wps:DataInputs/wps:Input[3]/ows:Identifier</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal></bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>wps:DataInputs/wps:Input[3]/ows:Title</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>
              <maps:InputLayerDescriptor>
    <maps:InputIdentifier>LayerToBeRated</maps:InputIdentifier>
    <maps:LayerDomain>
                <maps:CRS/>
    </maps:LayerDomain>
    <maps:LayerRange>
    <maps:NoDataValue>-9999</maps:NoDataValue>
    </maps:LayerRange>
</maps:InputLayerDescriptor>
            </bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>wps:DataInputs/wps:Input[3]/wps:ComplexValue/maps:InputLayerDescriptor</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>concat('EPSG:', $FloodSimulationRequest.request/AreaOfInterest/@crs )</bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>wps:DataInputs/wps:Input[3]/wps:ComplexValue/maps:InputLayerDescriptor/maps:LayerDomain/maps:CRS</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>OutputLayerDescriptor</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>wps:DataInputs/wps:Input[4]/ows:Identifier</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal></bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>wps:DataInputs/wps:Input[4]/ows:Title</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from part="request" variable="FloodSimulationRequest">
            <bpel:query>maps:OutputLayerDescriptor</bpel:query>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>wps:DataInputs/wps:Input[4]/wps:ComplexValue/maps:OutputLayerDescriptor</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>Unsigned8bit</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>wps:OutputDefinitions/wps:Output/@encoding</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>image/geotiff</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest">
            <bpel:query>wps:OutputDefinitions/wps:Output/@format</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
```

```
        <bpel:from>
          <bpel:literal>OUTLayer</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest">
          <bpel:query>wps:OutputDefinitions/wps:Output/ows:Identifier</bpel:query>
        </bpel:to>
      </bpel:copy>
    </bpel:assign>
    <bpel:invoke inputVariable="executeMASRequest" name="InvokeMAS" operation="execute" outputVariable="executeMASResponse"
partnerLink="WPSPL" portType="ps:ProcessingService"/>
    <bpel:assign name="PrepareResponse">
      <bpel:copy>
        <bpel:from part="return" variable="executeMASResponse"/>
        <bpel:to part="out" variable="FloodSimulationResponse"/>
      </bpel:copy>
    </bpel:assign>
    <bpel:reply operation="FloodSimulation" partnerLink="FloodSimulationPL" portType="peunha2:Peunha2" variable="FloodSimulationResponse"/>
  </bpel:sequence>
</bpel:process>
```

C.3 Damage Assessment Service

Graphical view and BPEL code follows.



```
<?xml version="1.0" encoding="UTF-8"?>
<!--Process Definitionusing ActiveBPEL(tm) Designer Version 4.0.0 (http://www.active-endpoints.com)
-->
<bpel:process xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
xmlns:das="http://localhost:8080/axis2/services/DamageAssessmentService"
xmlns:ext="http://www.activebpel.org/2006/09/bpel/extension/query_handling" xmlns:fas="http://eu-orchestra.org/OA/FeatureAccessService"
xmlns:fasTypes="http://eu-orchestra.org/OA/FeatureAccessService/types" xmlns:maps="http://maps.orchestra.jrc.it" xmlns:ns="http://eu-
orchestra.org/DamageAssessmentService/types" xmlns:ns1="Invalid Document" xmlns:oab="http://eu-orchestra.org/OA/OABasicService/types/1.0"
xmlns:ows="http://www.opengeospatial.net/ows" xmlns:peunha2="http://localhost:8080/axis2/services/Peunha2"
xmlns:ps="http://localhost:8080/axis2/services/ps" xmlns:pt="http://eu-orchestra.org/Peunha2/types" xmlns:tifrep="urn:service.orchestra.at"
xmlns:wps="http://www.opengeospatial.net/wps" xmlns:xsd="http://www.w3.org/2001/XMLSchema" ext:createTargetXPath="yes"
name="damageAssessment" suppressJoinFailure="yes" targetNamespace="http://wcs-test">
  <bpel:extensions>
    <bpel:extension mustUnderstand="yes" namespace="http://www.activebpel.org/2006/09/bpel/extension/query_handling"/>
  </bpel:extensions>
  <bpel:import importType="http://schemas.xmlsoap.org/wsdl/" location="http://naturegis.jrc.it/axis2-1.2/services/ProcessingService?wsdl"
namespace="http://localhost:8080/axis2/services/ps"/>
  <bpel:import importType="http://www.w3.org/2001/XMLSchema" location="WSDL/masInputTypes.xsd" namespace="http://maps.orchestra.jrc.it"/>
  <bpel:import importType="http://schemas.xmlsoap.org/wsdl/" location="WSDL/DamageAssessmentService.wsdl"
namespace="http://localhost:8080/axis2/services/DamageAssessmentService"/>
  <bpel:partnerLinks>
    <bpel:partnerLink name="WPSPL" partnerLinkType="das:WPSPLT" partnerRole="WPS"/>
    <bpel:partnerLink myRole="DamageAssessmentService" name="DamageAssessmentPL" partnerLinkType="das:DamageAssessmentPLT"/>
  </bpel:partnerLinks>
  <bpel:variables>
    <bpel:variable messageType="das:DamageAssessmentResponse" name="DamageAssessmentResponse"/>
    <bpel:variable name="ZonalRatingValueLayer_WCS_REF" type="xsd:string"/>
    <bpel:variable name="Width_ZonalRatingValueLayer" type="xsd:integer"/>
    <bpel:variable name="Height_ZonalRatingValueLayer" type="xsd:integer"/>
    <bpel:variable name="minX" type="xsd:decimal"/>
    <bpel:variable name="minY" type="xsd:decimal"/>
    <bpel:variable name="maxX" type="xsd:decimal"/>
    <bpel:variable name="maxY" type="xsd:decimal"/>
    <bpel:variable messageType="ps:executeRequest" name="executeMASRequest_ZonalRating"/>
    <bpel:variable messageType="ps:executeResponse" name="executeMASResponse_ZonalRating"/>
    <bpel:variable messageType="das:DamageAssessmentRequest" name="DamageAssessmentRequest"/>
    <bpel:variable name="Height_ZonalSumZonalLayer" type="xsd:integer"/>
    <bpel:variable name="Width_ZonalSumZonalLayer" type="xsd:integer"/>
    <bpel:variable name="ZonalSumZonalLayer_WCS_REF" type="xsd:string"/>
    <bpel:variable messageType="ps:executeRequest" name="executeMASRequest_ZonalSum"/>
    <bpel:variable messageType="ps:executeResponse" name="executeMASResponse_ZonalSum"/>
    <bpel:variable name="ZonalRatingResult_Ref" type="xsd:string"/>
    <bpel:variable messageType="ps:executeRequest" name="executeJoinRequest"/>
    <bpel:variable messageType="ps:executeResponse" name="executeJoinResponse"/>
    <bpel:variable name="AggregationLevel_WFS_REF" type="xsd:string"/>
    <bpel:variable name="ZonalSumResult_Ref" type="xsd:string"/>
    <bpel:variable messageType="ps:executeRequest" name="executeNormaliseRequest"/>
    <bpel:variable messageType="ps:executeResponse" name="executeNormaliseResponse"/>
    <bpel:variable name="NormaliseResult_Ref" type="xsd:string"/>
  </bpel:variables>
  <bpel:sequence>
    <bpel:receive createInstance="yes" name="ReceiveDamageAssessmentRequest" operation="DamageAssessment"
partnerLink="DamageAssessmentPL" portType="das:DamageAssessmentService" variable="DamageAssessmentRequest"/>
    <bpel:assign name="DeriveCoordinatesFromRequest">
      <bpel:copy>
        <bpel:from>number(substring-before( $DamageAssessmentRequest.request/BBOXEventExtent/LowerCorner,' '))</bpel:from>
        <bpel:to variable="minX"/>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>number(substring-before ( $DamageAssessmentRequest.request/BBOXEventExtent/UpperCorner, ' '))</bpel:from>
        <bpel:to variable="maxX"/>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>number(substring-after( $DamageAssessmentRequest.request/BBOXEventExtent/LowerCorner , ' '))</bpel:from>
        <bpel:to variable="minY"/>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>number(substring-after( $DamageAssessmentRequest.request/BBOXEventExtent/UpperCorner, ' '))</bpel:from>
        <bpel:to variable="maxY"/>
      </bpel:copy>
    </bpel:assign>
    <bpel:assign name="DeriveWidthAndHeight_4_PopDensityLayer_ZonalRatingValue">
      <bpel:copy>
        <bpel:from>round(&amp;#13;
(&amp;#13;(substring-before( $DamageAssessmentRequest.request/BBOXEventExtent/UpperCorner,' ')) -&amp;#13;( substring-before(
$DamageAssessmentRequest.request/BBOXEventExtent/LowerCorner,' '))&amp;#13;
) &amp;#13;0.0011635018)</bpel:from>
        <bpel:to variable="Width_ZonalRatingValueLayer"/>
      </bpel:copy>
```

```
      <bpel:copy>
        <bpel:from>round(&#13;
(&#13;(substring-after ($DamageAssessmentRequest.request/BBOXEventExtent/UpperCorner, ' ')) -&#13;(substring-after (
$DamageAssessmentRequest.request/BBOXEventExtent/LowerCorner, ' '))&#13;
) div 0.0011635018&#13;
)</bpel:from>
        <bpel:to variable="Height_ZonalRatingValueLayer"/>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>round(&#13;
(&#13;(substring-before( $DamageAssessmentRequest.request/BBOXEventExtent/UpperCorner,' ')) -&#13;( substring-before(
$DamageAssessmentRequest.request/BBOXEventExtent/LowerCorner,' '))&#13;
) &#13;0.027)</bpel:from>
        <bpel:to variable="Width_ZonalSumZonalLayer"/>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>round(&#13;
(&#13;(substring-after ($DamageAssessmentRequest.request/BBOXEventExtent/UpperCorner, ' ')) -&#13;(substring-after (
$DamageAssessmentRequest.request/BBOXEventExtent/LowerCorner, ' '))&#13;
) div 0.027&#13;
)</bpel:from>
        <bpel:to variable="Height_ZonalSumZonalLayer"/>
      </bpel:copy>
    </bpel:assign>
    <bpel:if name="ABSHeigth_ZonalRatingValueLayer">
      <bpel:condition>$Height_ZonalRatingValueLayer &lt; 0</bpel:condition>
      <bpel:assign name="SignChange">
        <bpel:copy>
          <bpel:from>- $Height_ZonalRatingValueLayer</bpel:from>
          <bpel:to variable="Height_ZonalRatingValueLayer"/>
        </bpel:copy>
      </bpel:assign>
    </bpel:if>
    <bpel:if name="ABSWidht_ZonalRatingValueLayer">
      <bpel:condition>$Width_ZonalRatingValueLayer &lt; 0</bpel:condition>
      <bpel:assign name="SignChange">
        <bpel:copy>
          <bpel:from>- $Width_ZonalRatingValueLayer</bpel:from>
          <bpel:to variable="Width_ZonalRatingValueLayer"/>
        </bpel:copy>
      </bpel:assign>
    </bpel:if>
    <bpel:assign name="PrepareWCS-Query_4_PopDensity_Exposure_ZonalRatingValueLayer">
      <bpel:copy>
        <bpel:from>concat (&#13;
'http://naturegis.jrc.it:8080/geoserver/wcs?REQUEST=GetCoverage&amp;Service=WCS&amp;version=1.0.0&amp;Coverage=orchestra:pop-density',
'&amp;format=GeoTIFF&amp;CRS=EPSG:', $DamageAssessmentRequest.request/BBOXEventExtent/@crs,'&amp;WIDTH=',
$Width_ZonalRatingValueLayer, '&amp;HEIGHT=', $Height_ZonalRatingValueLayer, '&amp;BBOX=', $minX, ',' , $minY, ',', $maxX, ',' , $maxY
)</bpel:from>
        <bpel:to variable="ZonalRatingValueLayer_WCS_REF"/>
      </bpel:copy>
    </bpel:assign>
    <bpel:assign name="prepareExecuteMASRequest_4ZonalRating">
      <bpel:copy>
        <bpel:from>
          <bpel:literal>WPS</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalRating">
          <bpel:query>@service</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>false</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalRating">
          <bpel:query>@status</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>true</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalRating">
          <bpel:query>@store</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
```

```
      <bpel:literal>ZonalRating</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeMASRequest_ZonalRating">
      <bpel:query>ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>0.4.0</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeMASRequest_ZonalRating">
      <bpel:query>@version</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>ZonalLayer</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeMASRequest_ZonalRating">
      <bpel:query>wps:DataInputs/wps:Input[1]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal></bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeMASRequest_ZonalRating">
      <bpel:query>wps:DataInputs/wps:Input[1]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal></bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeMASRequest_ZonalRating">
      <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from part="request" variable="DamageAssessmentRequest">
      <bpel:query>$DamageAssessmentRequest.request/EventLayer</bpel:query>
    </bpel:from>
    <bpel:to part="request" variable="executeMASRequest_ZonalRating">
      <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference/@ows:reference</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>ValueLayer</bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeMASRequest_ZonalRating">
      <bpel:query>wps:DataInputs/wps:Input[2]/ows:Identifier</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal></bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeMASRequest_ZonalRating">
      <bpel:query>wps:DataInputs/wps:Input[2]/ows:Title</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal></bpel:literal>
    </bpel:from>
    <bpel:to part="request" variable="executeMASRequest_ZonalRating">
      <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValueReference</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from variable="ZonalRatingValueLayer_WCS_REF"/>
    <bpel:to part="request" variable="executeMASRequest_ZonalRating">
      <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValueReference/@ows:reference</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>ZonalRatingMapping</bpel:literal>
```

```
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalRating">
          <bpel:query>wps:DataInputs/wps:Input[3]/ows:Identifier</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal></bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalRating">
          <bpel:query>wps:DataInputs/wps:Input[3]/ows:Title</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>
            <maps:ZonalRatingMapping valuesOutOfRangeBehaviour="NoDataValue">
    <maps:Mapping>
      <maps:ZoneLayerCombination>
        <maps:Zone>
          <maps:Value>0</maps:Value>
        </maps:Zone>
        <maps:Layer/>
      </maps:ZoneLayerCombination>
      <maps:OutputValue>0</maps:OutputValue>
    </maps:Mapping>
    <maps:Mapping>
      <maps:ZoneLayerCombination>
        <maps:Zone>
          <maps:Value>1</maps:Value>
        </maps:Zone>
        <maps:Layer/>
      </maps:ZoneLayerCombination>
      <maps:OutputValue>Layer</maps:OutputValue>
    </maps:Mapping>
</maps:ZonalRatingMapping>
          </bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalRating">
          <bpel:query>wps:DataInputs/wps:Input[3]/wps:ComplexValue/maps:ZonalRatingMapping</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>InputLayerDescriptorList</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalRating">
          <bpel:query>wps:DataInputs/wps:Input[4]/ows:Identifier</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal></bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalRating">
          <bpel:query>wps:DataInputs/wps:Input[4]/ows:Title</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy ignoreMissingFromData="yes">
        <bpel:from>
          <bpel:literal>
            <maps:InputLayerDescriptorList>
<maps:InputLayerDescriptor>
    <maps:InputIdentifier>ZonalLayer</maps:InputIdentifier>
    <maps:LayerDomain>
                <maps:CRS>EPSG:4326</maps:CRS>
    </maps:LayerDomain>
    <maps:LayerRange>
    <maps:NoDataValue>0</maps:NoDataValue>
    </maps:LayerRange>
</maps:InputLayerDescriptor>
<maps:InputLayerDescriptor>
    <maps:InputIdentifier>ValueLayer</maps:InputIdentifier>
    <maps:LayerDomain>
                <maps:CRS>EPSG:4326</maps:CRS>
    </maps:LayerDomain>
    <maps:LayerRange>
    <maps:NoDataValue>-9999</maps:NoDataValue>
    </maps:LayerRange>
```

```
</maps:InputLayerDescriptor>
</maps:InputLayerDescriptorList>
            </bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest_ZonalRating">
            <bpel:query>wps:DataInputs/wps:Input[3]/wps:ComplexValue/maps:InputLayerDescriptor</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>concat('EPSG:',$DamageAssessmentRequest.request/BBOXEventExtent/@crs )</bpel:from>
          <bpel:to part="request" variable="executeMASRequest_ZonalRating">
            <bpel:query>wps:DataInputs/wps:Input[3]/wps:ComplexValue/maps:InputLayerDescriptor/maps:LayerDomain/maps:CRS</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>OutputLayerDescriptor</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest_ZonalRating">
            <bpel:query>wps:DataInputs/wps:Input[4]/ows:Identifier</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal></bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest_ZonalRating">
            <bpel:query>wps:DataInputs/wps:Input[4]/ows:Title</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>
              <maps:OutputLayerDescriptor>
    <maps:LayerDomain>
      <maps:CRS>EPSG:4326</maps:CRS>
    </maps:LayerDomain>
      <maps:LayerRange>
      <maps:NoDataValue>-9999</maps:NoDataValue>
    </maps:LayerRange>
</maps:OutputLayerDescriptor>
            </bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest_ZonalRating">
            <bpel:query>wps:DataInputs/wps:Input[4]/wps:ComplexValue/maps:OutputLayerDescriptor</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>Unsigned16bit</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest_ZonalRating">
            <bpel:query>wps:OutputDefinitions/wps:Output/@encoding</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>image/geotiff</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest_ZonalRating">
            <bpel:query>wps:OutputDefinitions/wps:Output/@format</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>OUTLayer</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeMASRequest_ZonalRating">
            <bpel:query>wps:OutputDefinitions/wps:Output/ows:Identifier</bpel:query>
          </bpel:to>
        </bpel:copy>
    </bpel:assign>
    <bpel:invoke inputVariable="executeMASRequest_ZonalRating" name="InvokeMAPS_ZonalRating" operation="execute"
outputVariable="executeMASResponse_ZonalRating" partnerLink="WPSPL" portType="ps:ProcessingService"/>
    <bpel:assign name="Copy_ZonalRatingResultRef_AsVariable">
      <bpel:copy>
        <bpel:from part="return" variable="executeMASResponse_ZonalRating">
          <bpel:query>wps:ProcessOutputs/wps:Output/wps:ComplexValueReference/@ows:reference</bpel:query>
        </bpel:from>
```

```
        <bpel:to variable="ZonalRatingResult_Ref"/>
      </bpel:copy>
    </bpel:assign>
    <bpel:if name="ABSHeigthZonalSumLayer">
      <bpel:condition>$Height_ZonalSumZonalLayer &lt; 0</bpel:condition>
      <bpel:assign name="SignChange">
        <bpel:copy>
          <bpel:from>- $Height_ZonalSumZonalLayer</bpel:from>
          <bpel:to variable="Height_ZonalSumZonalLayer"/>
        </bpel:copy>
      </bpel:assign>
    </bpel:if>
    <bpel:if name="ABSWidht_ZonalSumLayer">
      <bpel:condition>$Width_ZonalSumZonalLayer &lt; 0</bpel:condition>
      <bpel:assign name="SignChange">
        <bpel:copy>
          <bpel:from>- $Width_ZonalSumZonalLayer</bpel:from>
          <bpel:to variable="Width_ZonalSumZonalLayer"/>
        </bpel:copy>
      </bpel:assign>
    </bpel:if>
    <bpel:assign name="PrepareWCS-Query_NUTS3AggregationLevel_4_ZonalSumZonalLayer">
      <bpel:copy>
        <bpel:from>concat (&#13;
'http://naturegis.jrc.it:8080/geoserver/wcs?REQUEST=GetCoverage&amp;Service=WCS&amp;version=1.0.0&amp;Coverage=orchestra:',
$DamageAssessmentRequest.request/AggregationLevel , '&amp;format=GeoTIFF&amp;CRS=EPSG:',
$DamageAssessmentRequest.request/BBOXEventExtent/@crs,'&amp;WIDTH=', $Width_ZonalSumZonalLayer, '&amp;HEIGHT=',
$Height_ZonalSumZonalLayer, '&amp;BBOX=', $minX, ',' , $minY, ',', $maxX, ',' , $maxY )</bpel:from>
        <bpel:to variable="ZonalSumZonalLayer_WCS_REF"/>
      </bpel:copy>
    </bpel:assign>
    <bpel:assign name="prepareExecuteMASRequest_4ZonalSum">
      <bpel:copy>
        <bpel:from>
          <bpel:literal>WPS</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalSum">
          <bpel:query>@service</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>0.4.0</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalSum">
          <bpel:query>@version</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>false</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalSum">
          <bpel:query>@status</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>true</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalSum">
          <bpel:query>@store</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>ZonalSum</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalSum">
          <bpel:query>ows:Identifier</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>ZonalLayer</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeMASRequest_ZonalSum">
          <bpel:query>wps:DataInputs/wps:Input/ows:Identifier</bpel:query>
        </bpel:to>
```

```
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal></bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeMASRequest_ZonalSum">
        <bpel:query>wps:DataInputs/wps:Input/ows:Title</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal></bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeMASRequest_ZonalSum">
        <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from variable="ZonalSumZonalLayer_WCS_REF"/>
      <bpel:to part="request" variable="executeMASRequest_ZonalSum">
        <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference/@ows:reference</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>ValueLayer</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeMASRequest_ZonalSum">
        <bpel:query>wps:DataInputs/wps:Input[2]/ows:Identifier</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal></bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeMASRequest_ZonalSum">
        <bpel:query>wps:DataInputs/wps:Input[2]/ows:Title</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal></bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeMASRequest_ZonalSum">
        <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValueReference</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from variable="ZonalRatingResult_Ref"/>
      <bpel:to part="request" variable="executeMASRequest_ZonalSum">
        <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValueReference/@ows:reference</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>8bit</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeMASRequest_ZonalSum">
        <bpel:query>wps:OutputDefinitions/wps:Output/@encoding</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>text/xml</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeMASRequest_ZonalSum">
        <bpel:query>wps:OutputDefinitions/wps:Output/@encoding</bpel:query>
      </bpel:to>
    </bpel:copy>
    <bpel:copy>
      <bpel:from>
        <bpel:literal>Table</bpel:literal>
      </bpel:from>
      <bpel:to part="request" variable="executeMASRequest_ZonalSum">
        <bpel:query>wps:OutputDefinitions/wps:Output/ows:Identifier</bpel:query>
      </bpel:to>
    </bpel:copy>
  </bpel:assign>
```

```
    <bpel:invoke inputVariable="executeMASRequest_ZonalSum" name="InvokeMAPS_ZonalSum" operation="execute"
outputVariable="executeMASResponse_ZonalSum" partnerLink="WPSPL" portType="ps:ProcessingService"/>
    <bpel:assign name="Copy_ZonalSumResultRef_AsVariable">
      <bpel:copy>
        <bpel:from part="return" variable="executeMASResponse_ZonalSum">
          <bpel:query>wps:ProcessOutputs/wps:Output/wps:ComplexValueReference/@ows:reference</bpel:query>
        </bpel:from>
        <bpel:to variable="ZonalSumResult_Ref"/>
      </bpel:copy>
    </bpel:assign>
    <bpel:assign name="prepareExecuteNormalise">
      <bpel:copy>
        <bpel:from>
          <bpel:literal>WPS</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>@service</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>0.4.0</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>@version</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>false</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>@status</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>true</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>@store</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>org.n52.wps.server.algorithm.NormalisationByFixedValueAlgorithm</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>ows:Identifier</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>InputFeatures</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>wps:DataInputs/wps:Input/ows:Identifier</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal></bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>wps:DataInputs/wps:Input/ows:Title</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal></bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from variable="ZonalSumResult_Ref"/>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference/@ows:reference</bpel:query>
```

```
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>NormaliseBy</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>wps:DataInputs/wps:Input[2]/ows:Identifier</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal></bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>wps:DataInputs/wps:Input[2]/ows:Title</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>100.0</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>wps:DataInputs/wps:Input[2]/wps:LiteralValue</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>xs:double</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>wps:DataInputs/wps:Input[2]/wps:LiteralValue/@dataType</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>Attributes</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>wps:DataInputs/wps:Input[3]/ows:Identifier</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal></bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>wps:DataInputs/wps:Input[3]/ows:Title</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>SUM</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>xs:string</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue/@dataType</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>URL_To_GMLResult</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeNormaliseRequest">
          <bpel:query>wps:OutputDefinitions/wps:Output/ows:Identifier</bpel:query>
        </bpel:to>
      </bpel:copy>
    </bpel:assign>
    <bpel:invoke inputVariable="executeNormaliseRequest" name="InvokeNormaliseService" operation="execute"
outputVariable="executeNormaliseResponse" partnerLink="WPSPL" portType="ps:ProcessingService"/>
    <bpel:assign name="Copy_NormaliseResultRef_AsVariable">
      <bpel:copy>
```

```
          <bpel:from part="return" variable="executeNormaliseResponse">
            <bpel:query>wps:ProcessOutputs/wps:Output/wps:LiteralValue</bpel:query>
          </bpel:from>
          <bpel:to variable="NormaliseResult_Ref"/>
        </bpel:copy>
      </bpel:assign>
      <bpel:assign name="PrepareWFSQuery_REF_4_AggregationLevel">
        <bpel:copy>
          <bpel:from>concat
('http://naturegis.jrc.it:8080/geoserver/wfs?request=GetFeature&amp;version=1.0.0&amp;service=wfs&amp;typeName=orchestra:',
$DamageAssessmentRequest.request/AggregationLevel , '&amp;SRS=EPSG:', $DamageAssessmentRequest.request/BBOXEventExtent/@crs,
'&amp;BBOX=', $minX, ',' , $minY, ',', $maxX, ',' , $maxY )</bpel:from>
          <bpel:to variable="AggregationLevel_WFS_REF"/>
        </bpel:copy>
      </bpel:assign>
      <bpel:assign name="prepareExecuteJoin">
        <bpel:copy>
          <bpel:from>
            <bpel:literal>WPS</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeJoinRequest">
            <bpel:query>@service</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>0.4.0</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeJoinRequest">
            <bpel:query>@version</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>false</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeJoinRequest">
            <bpel:query>@status</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>true</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeJoinRequest">
            <bpel:query>@store</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>org.n52.wps.server.algorithm.LeftOuterJoinAlgorithm</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeJoinRequest">
            <bpel:query>ows:Identifier</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal>LeftFeatures</bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeJoinRequest">
            <bpel:query>wps:DataInputs/wps:Input[1]/ows:Identifier</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal></bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeJoinRequest">
            <bpel:query>wps:DataInputs/wps:Input[1]/ows:Title</bpel:query>
          </bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from>
            <bpel:literal></bpel:literal>
          </bpel:from>
          <bpel:to part="request" variable="executeJoinRequest">
            <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference</bpel:query>
          </bpel:to>
```

```
</bpel:copy>
<bpel:copy>
  <bpel:from variable="AggregationLevel_WFS_REF"/>
  <bpel:to part="request" variable="executeJoinRequest">
    <bpel:query>wps:DataInputs/wps:Input[1]/wps:ComplexValueReference/@ows:reference</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal>JoinAttributeLeftCollection</bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeJoinRequest">
    <bpel:query>wps:DataInputs/wps:Input[2]/ows:Identifier</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal></bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeJoinRequest">
    <bpel:query>wps:DataInputs/wps:Input[2]/ows:Title</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal></bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeJoinRequest">
    <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValueReference</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from variable="NormaliseResult_Ref">
    <bpel:query>$NormaliseResult_Ref</bpel:query>
  </bpel:from>
  <bpel:to part="request" variable="executeJoinRequest">
    <bpel:query>wps:DataInputs/wps:Input[2]/wps:ComplexValueReference/@ows:reference</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal>JoinAttributeLeftCollection</bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeJoinRequest">
    <bpel:query>wps:DataInputs/wps:Input[3]/ows:Identifier</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal></bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeJoinRequest">
    <bpel:query>wps:DataInputs/wps:Input[3]/ows:Title</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal>CODE</bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeJoinRequest">
    <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal>xs:string</bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeJoinRequest">
    <bpel:query>wps:DataInputs/wps:Input[3]/wps:LiteralValue/@dataType</bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from>
    <bpel:literal>JoinAttributeRightCollection</bpel:literal>
  </bpel:from>
  <bpel:to part="request" variable="executeJoinRequest">
    <bpel:query>wps:DataInputs/wps:Input[4]/ows:Identifier</bpel:query>
  </bpel:to>
</bpel:copy>
```

```
      <bpel:copy>
        <bpel:from>
          <bpel:literal></bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeJoinRequest">
          <bpel:query>wps:DataInputs/wps:Input[4]/ows:Title</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>ZONE</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeJoinRequest">
          <bpel:query>wps:DataInputs/wps:Input[4]/wps:LiteralValue</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>xs:string</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeJoinRequest">
          <bpel:query>wps:DataInputs/wps:Input[4]/wps:LiteralValue/@dataType</bpel:query>
        </bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from>
          <bpel:literal>URL_To_GMLResult</bpel:literal>
        </bpel:from>
        <bpel:to part="request" variable="executeJoinRequest">
          <bpel:query>wps:OutputDefinitions/wps:Output/ows:Identifier</bpel:query>
        </bpel:to>
      </bpel:copy>
    </bpel:assign>
    <bpel:invoke inputVariable="executeJoinRequest" name="InvokeJoinService" operation="execute" outputVariable="executeJoinResponse"
partnerLink="WPSPL" portType="ps:ProcessingService"/>
    <bpel:assign name="PrepareResponse">
      <bpel:copy>
        <bpel:from part="return" variable="executeJoinResponse"/>
        <bpel:to part="out" variable="DamageAssessmentResponse"/>
      </bpel:copy>
    </bpel:assign>
    <bpel:reply operation="DamageAssessment" partnerLink="DamageAssessmentPL" portType="das:DamageAssessmentService"
variable="DamageAssessmentResponse"/>
  </bpel:sequence>
</bpel:process>
```

Appendix D: The Raster Shift Problem

D.1 Rules for raster spaces and their representation

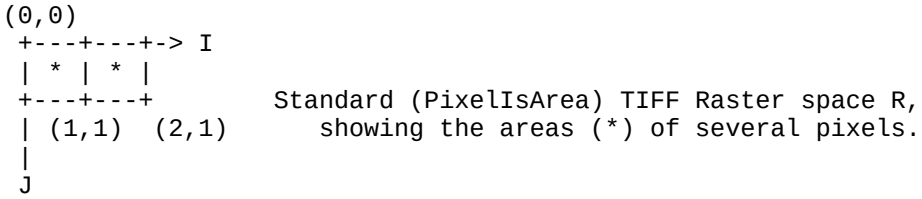
The following is taken from http://www.remotesensing.org/geotiff/spec/geotiff2.5.html.

"PixelIsArea" Raster Space

The "PixelIsArea" raster grid space R, which is the default, uses coordinates I and J, with (0,0) denoting the upper-left corner of the image, and increasing I to the right, increasing J down. The first pixel-value fills the square grid cell with the bounds:

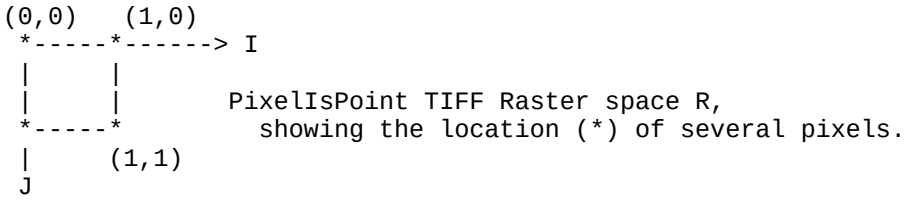
top-left = (0,0), bottom-right = (1,1)

and so on; by extension this one-by-one grid cell is also referred to as a pixel. An N by M pixel image covers an area with the mathematically defined bounds (0,0),(N,M).



"PixelIsPoint" Raster Space

The PixelIsPoint raster grid space R uses the same coordinate axis names as used in PixelIsArea Raster space, with increasing I to the right, increasing J down. The first pixel-value however, is realized as a point value located at (0,0). An N by M pixel image consists of points which fill the mathematically defined bounds (0,0),(N-1,M-1).



The PixelIsPoint representation is used in order to represent one particular value per point. This is sensible e.g. for radar data for digital elevation models, or for the representation of sensor networks where each point represents a value. How the values are interpreted and displayed (i.e. interpolation method) is not fixed within the header. Conceptually the PixelIsPoint is quite different from the PixelIsArea representation, which is generally used for raster images created by optic sensors.

If a PixelIsPoint representation were to be displayed on a display device with pixel cells having the same size as the raster spacing, then the upper-left corner of the displayed image would be located in raster space at (-0.5, -0.5). This option makes use of the Voronoi interpolation in its most simple form (http://en.wikipedia.org/wiki/Voronoi_diagram).

D.2 Experiments

1) CREATION OF SAMPLE FILE IN ERDAS

Creating a file with the following characteristics:

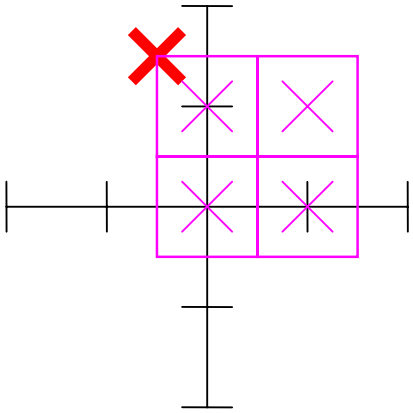
Tie point: - 0.5 / 1.5

Step width: 1, 1

Representation: PixelIsArea

BoundingBox:
<gml:pos>-0.5 -0.5</gml:pos>
<gml:pos>1.5 1.5</gml:pos>

Interpretation of the representation in Erdas and ArcGIS:



==========

2) IMPORT / EXPORT IN GEOSERVER

Reading this into GeoServer, which transforms the file into the representation PixelIsPoint and requesting the file through WCS interface:

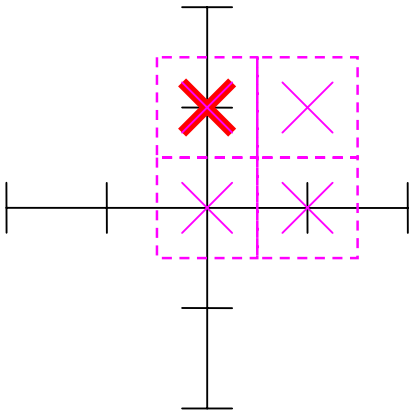
Tie point: 0.0 / 1.0

Step width: 1, 1

Representation: PixelIsPoint

BoundingBox:
<gml:pos>-0.5 -0.5</gml:pos>
<gml:pos>1.5 1.5</gml:pos>

Interpretation of the representation in Erdas and ArcGIS (using Verona-Polygons):



Discussion: The fact that GeoServer is doing this can be considered part of the problem. The raster representation is chosen for good reason. By changing it, we adopt a model that is **inadequate** for Map Algebra which works with area representations (particularly when performing zonal operations).

3) IMPORT / PROCESSING / EXPORT IN GRASS

Reading, processing and exporting data coming from GeoServer, using GRASS and the GDAL libraries. We had experimented that already the import produces this shift.

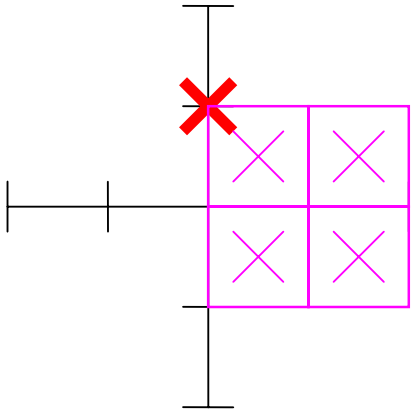
Tie point: 0.0 / 1.0

Step width: 1, 1

Representation: PixelIsArea **ERROR**

BoundingBox:
<gml:pos>0.0 -1.0</gml:pos>
<gml:pos>2.0 1.0</gml:pos>

Interpretation of the representation in Erdas and ArcGIS (using Voronoi-Polygons):



Discussion: The fact that it overwrites the representation PixelIsPoint with PixelIsArea seems to imply that GRASS ignores the information on the representation. However, there is the slight chance that this is not considered as a bug: The interpretation of the difference in the representation of is an open issue in the GeoTIFF/Coverage community.

D.3 Conclusion

There are several solutions that could be adopted in order to prevent the shift. One example is editing the GeoTIFF header. However, doing so would affect the performance of the service as it requires additional reading and writing activities of possibly large binary datasets.

Furthermore, the problem should be solved by the geotiff/Grid Coverage community or the used tools, i.e.

1. The geotiff/coverage community has to align the way how the geotiff standard should be interpreted.
2. Geoserver should not return all GeoTiff as PixelIsPoint. The Coverage creator should be able to decide what he wants to serve, and conceptually the two types of representation are very different.
3. GRASS should not assume that anything it receives is PixelIsArea, or, as a matter of fact, interpret their tie points in the same way (see point 1).

European Commission
EUR 24001 – Joint Research Centre – Institute for Environment and Sustainability

Title: **Pan-European Risk Management in SDIs**

Editor(s): Nicole Ostlaender, Roberto Lucchi, Michael Lutz, Anders Friis-Christensen, Ioannis Kannellopoulos, Angelo Quaglia

Luxembourg: Publications Office of the European Union

2013 – 150 pp. – 21.0 x 29.7 cm

EUR – Scientific and Technical Research series – ISSN 1831-9424 (online)

ISBN 978-92-79-19091-9 (pdf)

doi:10.2788/37971

Abstract

This report describes the ORCHESTRA pilot for pan-European hazard-assessment (PEUNHA). ORCHESTRA stands for Open Architecture and Spatial Data Infrastructure for Risk Management. As an Integrated Project, ORCHESTRA was partly funded by the European Commission's 6th framework program within the action IST-2002-2.3.2.9 Improving Risk management [IST-2002-51167]. The goal of ORCHESTRA was to design and implement the specifications for a service oriented spatial data infrastructure to improve interoperability between risk management authorities in Europe, and to improve the handling of disaster risk reduction strategies and emergency management operations. This report describes the work on a pilot for pan-European hazard-assessment, carried out during the ORCHESTRA Work Package (WP) 4.2. The focus of the pilot lies on the development of applications supporting hazard, damage and risk assessment for forest fires and floods. The report covers the application and related use cases, the platform-specific service specifications, and the prototypic implementations. It closes with lessons learned.

LB-NA-24001-EN-N

As the Commission's in-house science service, the Joint Research Centre's mission is to provide EU policies with independent, evidence-based scientific and technical support throughout the whole policy cycle.

Working in close cooperation with policy Directorates-General, the JRC addresses key societal challenges while stimulating innovation through developing new standards, methods and tools, and sharing and transferring its know-how to the Member States and international community.

Key policy areas include: environment and climate change; energy and transport; agriculture and food security; health and consumer protection; information society and digital agenda; safety and security including nuclear; all supported through a cross-cutting and multi-disciplinary approach.

Publications Office

ISBN 978-92-79-19091-9

9 789279 190919